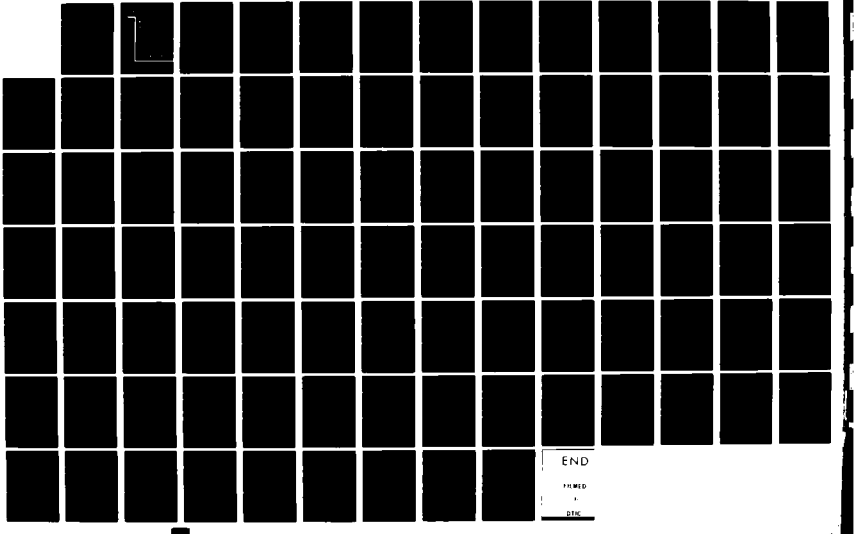
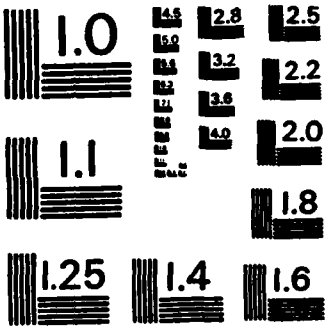
AD-A121 017 MICROTERMINAL/MICROFICHE TESTING SYSTEM(U) DENVER RESEARCH INST COLO ELECTRONICS DIV J P LAMOS ET AL. SEP 82 AFHRL-TP-82-17 F33615-78-C-0046 1/1

UNCLASSIFIED F/G 5/9 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFHRL-TP-82-[illegible]00 80 11 28

(10)

AIR FORCE HUMAN RESOURCES LABORATORY

AD A121017

MICROTERMINAL/MICROFICHE TESTING SYSTEM

By

Joseph P. Lamos
A. E. Winterbauer

Electronics Division
Denver Research Institute
University of Denver
Denver, Colorado 80208

LOGISTICS AND TECHNICAL TRAINING DIVISION
Technical Training Branch
Lowry Air Force Base, Colorado 80230

DTIC ELECTE NOV 3 198

September 1982

Final Technical Paper

Approved for public release; distribution unlimited.

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

DTIC FILE COPY

82 11 03 007

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

BRIAN DALLMAN
Contract Monitor

JOSEPH A. BIRT, LtCol, USAF
Technical Director, Logistics and Technical Training Division

DONALD C. TETMEYER, Colonel, USAF
Chief, Logistics and Technical Training Division

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFHRL-TP-82-17	2. GOVT ACCESSION NO. A121017	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* MICROTERMINAL/MICROFICHE TESTING SYSTEM		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Joseph P. Lamos A. E. Winterbauer		8. CONTRACT OR GRANT NUMBER(s) F33615-78-C-0046
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electronics Division, Denver Research Institute University of Denver Denver, Colorado 80208		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62205F 11210901
11. CONTROLLING OFFICE NAME AND ADDRESS HQ Air Force Human Resources Laboratory (AFSC) Brooks Air Force Base, Texas 78235		12. REPORT DATE September 1982 13. NUMBER OF PAGES 90
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* Logistics and Technical Training Division Technical Training Branch Air Force Human Resources Laboratory Lowry Air Force Base, Colorado 80230		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

advanced instructional system
computer-assisted instruction
computer-based instruction
computer-based testing
computer-managed instruction
interactive testing
measurement
microfiche
microterminal
progression testing
selective testing
technical training
test management

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Microterminal/Microfiche system is the result of a three-phase effort toward establishing a stand-alone, testing device to complement technical training in a computer-based instructional (CBI) environment. The objectives in developing such a system were to interface existing Air Force microterminals with low-cost, commercially available microfiche readers via computer hardware and software in order to provide the capability for using microfiche as a testing medium; to develop computer hardware and software interfaces between a grading microterminal and a microcomputer; and to develop microcomputer software; all of which would facilitate test management and administrative functions. The system supports progression testing (i.e., the serial presentation of criterion-referenced

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

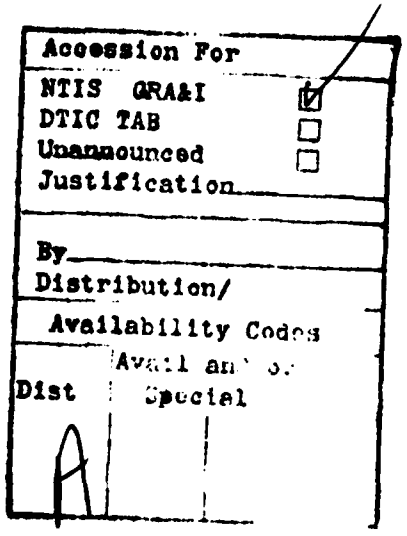
Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Item 20 (Continued):

test items), as well as selective testing (i.e., the ability to retest students over similar material with a different testing algorithm).

Because of the interfacing capability of the microterminal and its data-input characteristics, the system is flexible enough for consideration to be given to the microterminal's role as something other than a testing device. System configurations could include videodisk players, as well as filmstrips, slides, etc., for presentation media synchronized with the response handling capability of the microterminal and at considerable savings relative to other CBI systems.



AFHRL Technical Paper 82-17 September 1982

MICROTERMINAL/MICROFICHE TESTING SYSTEM

By

Joseph P. Lamos
A. E. Winterbauer

Electronics Division
Denver Research Institute
University of Denver
Denver, Colorado 80208

Reviewed by

Joseph Y. Yasutake
Senior Scientist, Technical Training Branch

Submitted for publication by

Allen J. Partin, Lt Col, USAF
Chief, Technical Training Branch

SUMMARY

The Microterminal/Microfiche (MT/MF) system is the result of a three-phase effort toward establishing a stand-alone, testing device to complement technical training in a computer-based instructional (CBI) environment. The objectives in developing such a system were to interface existing Air Force microterminals with low-cost, commercially available microfiche readers via computer hardware and software in order: (a) to provide the capability for using microfiche as a testing medium, (b) to develop computer hardware and software interfaces between a grading microterminal and a microcomputer, and (c) to develop microcomputer software, all of which would facilitate test management and administrative functions. The system supports progression testing (i.e., the serial presentation of criterion-referenced test items), as well as selective testing (i.e., the ability to retest students by selectively administering test items from a test item pool).

BACKGROUND

A CBI system is composed of computer-assisted instruction (CAI) and computer-managed instruction (CMI). If the CBI system contains a large CAI component, the costs of the dedicated terminals for the students can become very high. The total cost of CMI terminals, on the other hand, is not as expensive because they are management tools and as a consequence their numbers are based on increments in the student population. This is contrasted to having to provide a CAI terminal for each student who is to be given CAI. CMI, however, lacks interaction at the lesson level and does have additional costs associated with the need for computer forms and associated support materials.

The MT/MF system provides low-cost student terminals, as well as a microcomputer to handle CMI that is interactive at the lesson level. Information can be presented, responses can be handled, and data can be collected in an inexpensive configuration.

APPROACH

Phase I efforts were directed toward interfacing the student microterminal to a manual-access microfiche reader to provide automatic tracking between the two devices of the testing material. The results of Phase I are presented in Kottenstette, Steffen and Lamos (1980).

In order to provide selective testing capability, as well as test management and administrative functions, Phases II and III were directed toward developing an interface between the grading microterminal and a microcomputer to provide microcomputer software control of course- and student-management procedures. To facilitate downloading from and uploading into a microcomputer memory buffer for the exchange of student

data and testing procedures, a serial interface assembly conforming to RS-232C specifications was designed and installed in the existing microterminal used by the Air Force. Microcomputer software written in UCSD PASCAL was designed to provide CMI capability down to the lesson level.

SPECIFICS

An RS-232C serial interface board was designed, built, and installed in the grading microterminal to allow the testing data contained in a student's memory module to be uploaded into a buffer in a microcomputer with RS-232 interface capability. Software was designed to grade this material, to store the results in appropriate files, and to reinitialize the module for future student testing. Appropriate checking procedures were incorporated into the software to preclude student error, such as attempting to take the wrong test.

Software was also designed for the microcomputer to facilitate such CMI functions as establishing courses and versions of courses, testing blocks, student group enrollment and assignment to the appropriate course and block, student progress, and the generation of appropriate reports to allow student, course, and test evaluation. The system can also aid the instructor/course designer in establishing test formats.

RESULTS AND CONCLUSIONS

The result of this and preceding contract work has successfully demonstrated the development of an alternate CBI technology. The microterminal, in combination with the memory module, has potential as a low-cost, data-collection device capable of distributing data. The microcomputer component, when interfaced with the microterminal, provides stand-alone CMI/Test Administration and Evaluation capability. The MT/MF system has the potential for microfiche-based testing, given proper production facilities, and, therefore, could contribute toward the reduction of paperwork and enhance test security. As it is, tests could be generated quite simply through the use of any of the microcomputer, word-processing software packages commercially available.

Because of the RS-232C interfacing capability of the microterminal and its data-input characteristics, the system is flexible enough for consideration to be given to the microterminal's role as something other than a testing device. System configurations could include RS-232 compatible videodisk players, as well as filmstrips, slides, etc., for presentation media synchronized with the response handling capability of the microterminal and at considerable savings relative to other CBI systems.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The development of the Air Force Human Resources Laboratory (AFHRL) Microterminal/Microfiche (MT/MF) System was planned for three phases. Phase I is documented in Kottenstette, et al. (1980). The present paper documents Phases II and III. Phase II efforts were devoted to developing an interface between the student microterminal and a microcomputer to facilitate the exchange of student data, test data, and testing procedures. To complete the mission of establishing stand-alone capability for the MT/MF system, Phase III was directed toward (a) the development of Test Administration and Delivery software to utilize the microterminal/microcomputer hardware interface created in Phase II, thereby providing selective-testing capability, (b) demonstrating the operation of the combined units, and (c) providing a Users Manual and a Programmers Manual for personnel operating the hardware and software developed during Phases II and III.

A. BACKGROUND

The activities of Phases II and III, detailed in this technical paper, represent the culmination of several years of effort exploring the design and use of low-cost, alternative-delivery technology in computer-based instruction. The context for the effort was the Air Forces Advanced Instructional System (AIS). The AIS combined both computer-managed instruction (CMI) and computer-assisted instruction (CAI) in a centralized computer-based instruction (CBI) system. The AIS supported four major technical training courses at Lowry AFB: (a) the Precision Measurement Equipment Course, (b) the Inventory Management Course, (c) the Materiel Facilities Course, and (d) the Weapons Mechanics Course. These courses were given in three different buildings at Lowry AFB, with CMI and CAI capabilities provided from the central AIS computer facility. These four courses represented approximately 21% of the student body in the Lowry Technical Training Center. The capacity of the AIS was geared to the management of over 2,000 students on a daily basis.

From the beginning of the AIS program, the major emphasis of the development of AIS technology was on the provision of CMI, since CMI was seen to be the more cost-effective component of CBI for the large number of students and course content that would be supported by the system. Additionally, at the time the AIS program was initiated in 1973, the cost of CAI delivery was considerable. A significant aspect of the cost at that time were CAI terminals and the communication costs of an on-line network. The cost of providing CAI terminals is a linear function of the number of students supported on-line at one time, regardless of whether the system is time-shared or, as is now available, the CAI is provided by a networked set of intelligent, stand-alone terminals. On the other hand, CMI terminal costs are a stepwise

function of the number of students supported; each node of the CMI terminal capability is able to support a range of "on-line" student numbers. Within a given range, increases in the student capacity can be accommodated without necessitating an increase in the number of management terminals.

The initial terminals which the AIS supported were of two types. The first was called an "A" terminal, which was a plasma-type terminal patterned after the PLATO IV terminal; the "A" terminal was the CAI terminal. The second type was the management terminal and was referred to as the "B" terminal. This terminal consisted of a full-page, optical-mark reader, a printer, and a Digital Equipment Corporation model PDP 11/05 minicomputer, functioning as a local processor. These two types of terminals served as the reference point for the consideration and development of alternative terminal options within the AIS context.

As has been mentioned, CMI is particularly cost-effective as compared to CAI because its capabilities can be distributed over a large number of students and off-line instructional content. CMI improves the efficiency of the instructional enterprise because it permits easy administration of a diverse set of instructional options (e.g., different media or instructional strategies) in accordance with the individual needs of the students. In this sense, CAI can be regarded as a particular type of instructional medium, managed along with other media types, such as printed text.

One major administrative component of CMI is the testing and evaluation of students. In the AIS, frequent student testing was necessary because of the desire to carefully prescribe instruction on an adaptive basis to the individual needs of the students. Since 90% of testing was planned to be performed via off-line, non-CAI materials, thousands of computer-readable test forms were being used each year. At an approximate cost of 3.5 cents each, the cost of forms over a 5-year period, with maximum student load on the system, would be considerable. Additionally, since computer test forms were read in through an optical mark reader, a basically mechanical device, reliability also became a concern because of alignment errors and errors due to the density of student marks. These concerns led AFHRL to consider potential alternatives to the "B" terminal and to computer forms for the recording of student test input.

The first effort for determining an alternative test-input device was accomplished in-house by AFHRL staff personnel. This effort (Kirby and Gardner, 1976) resulted in a breadboard prototype testing device. The Denver Research Institute (DRI) took the results of this initial effort and, in a separate contract effort, built a refined set of devices and evaluated their use to support test administration in the AIS. The results of this effort were reported in AFHRL-TR-78-50,

(Steffen, Gray, Wasmundt, and Lamos, 1978). This device, known as the microterminal, is shown in Figure 1.

The microterminal is a desk-top unit which is characterized by its simplified display capabilities, 16-key keypad, and plug-in memory module. These features were devised in response to explicit design criteria for achieving a low-cost device to be used in conjunction with off-line materials. The display features of the microterminal combine a four-place, alphanumeric, LED-readout unit with a series of nine LED indicators for identifying the status of predetermined system input conditions or messages such as "ENTER YOUR SSN" or "ENTER YOUR BOOKLET NUMBER". The number of keys on the micro-terminal is sufficient to permit the answering of objective-type test items, and special function keys allow a student to move through a test in a manner equivalent to using a test form and a pencil; i.e., a student using this microterminal is capable of skipping items, which are automatically tracked so the student can return to them with a single key press or can jump to any particular item to change the answer or to review the answer selected.

The plug-in memory module for the microterminal is an important aspect of the original design effort and the development effort conducted in Phases I to III of the present effort. This memory module contains random access memory (RAM) which is sustained by its own internal, rechargeable-battery source. With the memory module plugged into the microterminal, the student's test responses are stored and retained until the memory module's information is down-loaded to a computer for test grading, updating of the test and student records, and the issuing of a student assignment report. For the previous effort, the centralized computer was the AIS computer, and memory module information was down-loaded through an interface with the local processor in the AIS management terminal. In this sense, the memory module became an electronic test "form," a "form" which was capable of immediate processing or storing for later processing.

The use of the microterminal in the AIS classroom environment provided evidence that the speed and accuracy with which students could complete a test were improved and that the logistics and costs associated with test administration in a CBI system could also be improved. The results of this effort led the Air Force to consider several additional capabilities for the use of the microterminal technology to achieve cost-effectiveness in the AIS CBI context. These considerations resulted in the three phases of the present contract effort.

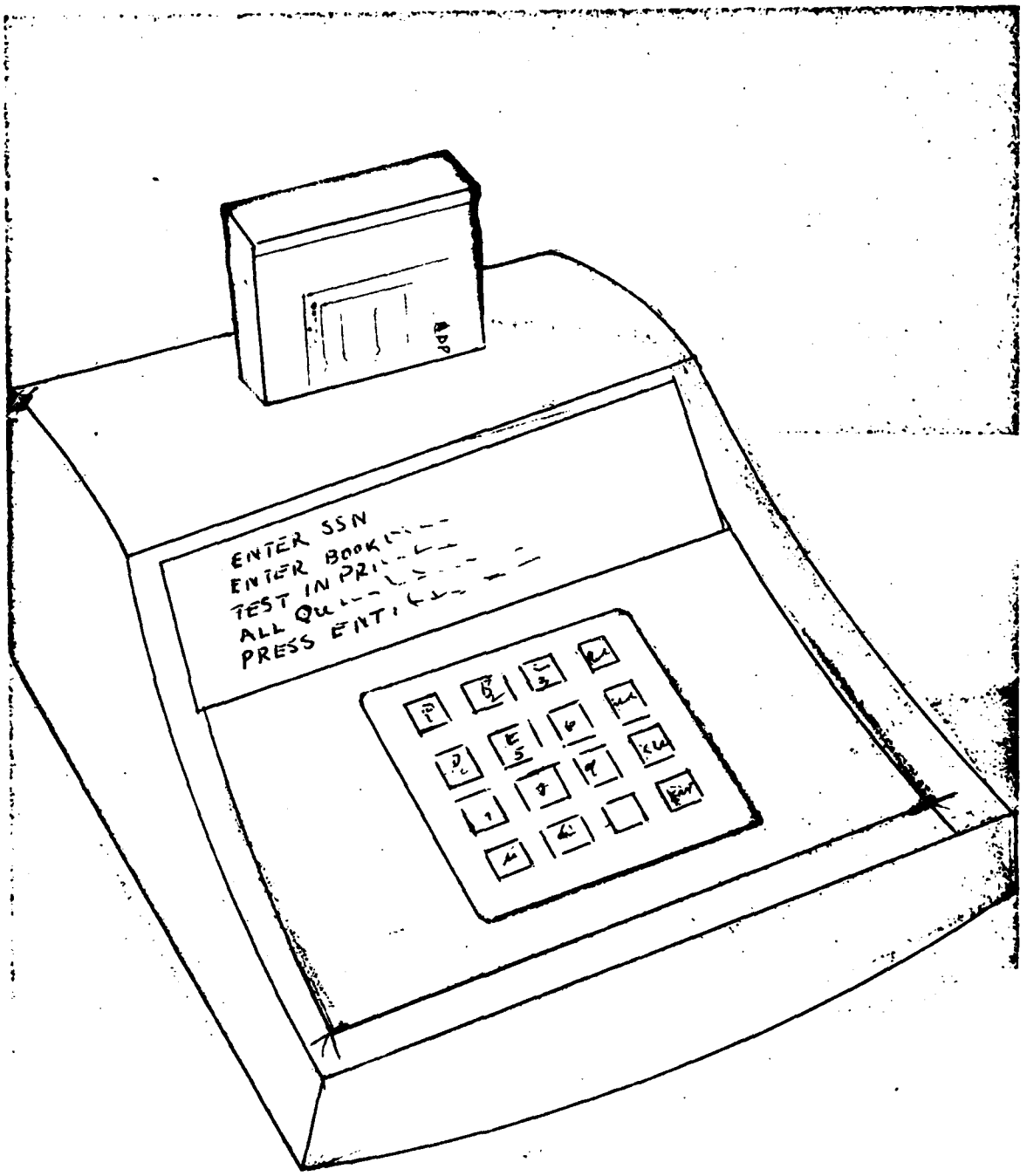


Figure 1. Student Microterminal

B. PHASE I ACTIVITIES

In addition to providing a reliable and effective mechanism for recording test answers, the Air Force was concerned with improving test security. The large number of paper test booklets needed in the AIS courses created a logistics problem in keeping tests secure. With paper test booklets, security was achieved by maintaining a locked test cabinet and having a stringent set of signout procedures. Phase I of the present study explored how the microterminal technology could be used for improving the security of test administration.

The obvious solution to improving test security would be to encode test items in a medium which inherently provides a mechanism of controlled access. One way of achieving this controlled access would be to present test items via a CAI terminal, thereby achieving computer-assisted testing (CAT). However, with typical CAI terminals or with stand-alone microcomputer systems capable of showing text and medium-resolution graphics (in technical training, many test items require accompanying technical illustrations), delivery costs would be increased dramatically over those of the existing paper-based medium. Since the microterminal already provides a highly flexible and secure way of recording test answers, it was determined that linking the microterminal to a more secure medium of presenting information could provide a cost-effective solution. Because previous research by AFHRL shows that microfiche could be used as an effective instructional medium (Kottenstette, 1979) and because microfiche is a more secure medium for testing situations, it was decided to explore the development of a low-cost capability for interfacing the presentation of test items on microfiche with the control and response capabilities of the microterminal.

As a consequence, Phase I activities resulted in a test-administration system known as the MT/MF system. The MT/MF system was developed by taking an off-the-shelf, manual-access, microfiche reader and adding an electronic detection system to the platen of the reader and a control for the projection lamp of the reader. This detection system was then interfaced with the microterminal in such a way that the microterminal could recognize the exact location on the microfiche where the student was looking. This approach was selected because automatic microfiche readers were too expensive for consideration, and because it was felt that the manual location of a test item on a single fiche was not a cumbersome task for the student to perform. A single fiche, depending on the reduction size chosen, could provide storage for at least 126 test items. The ability to sense the fiche location where the student was looking and control the lamp precluded the chance of the students answering the wrong test item.

For the Phase I MT/MF system, the software program embedded in the microterminal interpreted the booklet number entered by a student to determine the particular test the student would be given. In the AIS courses, each test usually had two to three alternative forms. For the

evaluation of the MT/MF system, three alternatives of a particular test were placed on a single fiche. After the student entered a booklet number, the microterminal, through the microfiche reader interface, activated the light source in the reader only when the student had the reader's platen in the correct area for the test assigned. Thus, the student was prevented from looking at any other version of the test. The other tests could be assigned later if the student failed the first assigned test.

The results of Phase I showed that the microterminal technology could be effectively used to control microfiche presentation of test items. The majority of students who used the MT/MF system experienced no difficulty in following the test messages on the microterminal, nor did they have any difficulty in manually locating test items on the fiche.

C. PROJECT HISTORY OF PHASE II

The impetus behind Phase II was to establish an MT/MF system capacity for selective testing; that is, the ability to test different students or retest one student by utilizing a pool of test items which were placed on fiche, and different testing algorithms which were controlled by software. To provide such a capability, an interface was developed for the memory module which consisted of a standard RS-232C with ASCII protocol that would allow communication with a large variety of computer equipment. A detailed description of the interface is given in Section II of this report. Such an interface, in conjunction with controlling software in the microterminal, allowed the memory module to be down-loaded with data from an external source (a microcomputer or CBI system) to provide a variety of testing algorithms to control the presentation of unique test item sequences on the microfiche viewer. The interface was developed in a manner that would allow the MT/MF system to function with any CBI system. However, to simplify the demonstration of the selective testing capability, a stand-alone microcomputer was used to generate the necessary data that could be down-loaded to the memory module in place of a larger central computer, such as the one used in the AIS.

A secondary purpose for using a microcomputer was to provide baseline information for determining how such a configuration (small computer and microterminals) would function in a stand-alone mode to completely support a student testing center in instructional environments where no other CBI systems exist. Sections II and III present in more detail the hardware and software efforts of Phase II.

D. PROJECT HISTORY OF PHASE III

The success of Phase II stimulated a re-direction of the effort during Phase III from that of enhancing the MT/MF system capability to perform with an existing larger CBI system to an extended software development effort for the microterminal and microcomputer to create a stand-alone, student-testing facility.

This effort consisted of:

1. Finalizing the interfacing of the microterminal to a central facility (microcomputer) for the support of test administration and test delivery.

2. Providing a capability in the central microcomputer for the storage or generation of test keys.

3. Developing a "test editor" in the central microcomputer for the production of test keys and the creation of a student test record data base to effect a complete stand-alone, computer-based testing facility for use in either group or self-paced resident training courses.

4. Completing the engineering drawings and related documentation for all hardware developed during all phases of the contract.

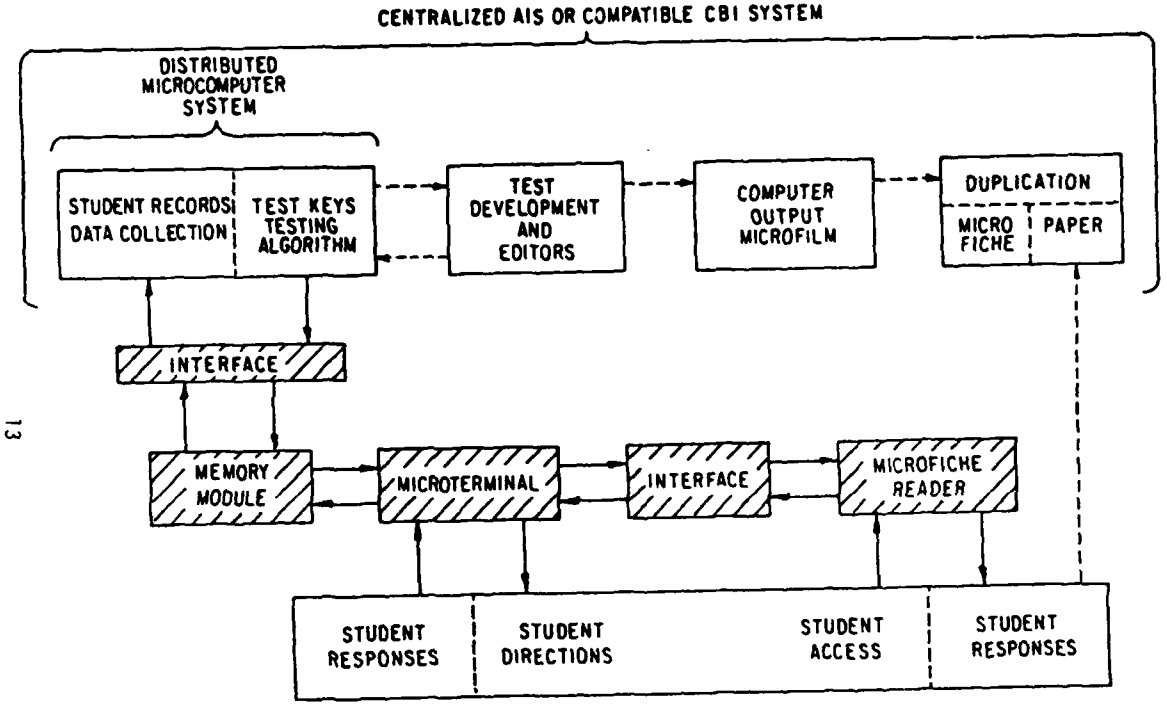
E. MICROTERMINAL/MICROFICHE SYSTEM DESCRIPTION

The MT/MF system is composed of five major components and associated interfaces: (a) microterminal, (b) microfiche reader, (c) memory module, (d) hardware interface between the microterminal and the microfiche reader, and (e) hardware and software interfaces between the microterminal and the central microcomputer. The hardware interface between the microterminal and the microfiche reader enables the coordination of the information presented on the microfiche with the instructional strategy implemented and controlled by the microterminal. In addition, communication with the microcomputer or other CBI system is required to permit data transfer to and from the microterminal memory module. A Test Administration and Delivery software package is necessary to allow the MT/MF system to exist as a stand-alone system, separate from any mainframe CBI system, such as the AIS.

The MT/MF system is capable of complementing CBI systems, such as the AIS, by providing for: (a) student response handling (microterminal) (b) test presentation (microfiche), and (c) data collection relative to the student's transactions on the MT/MF (memory module). The relationship between the MT/MF components and the microcomputer (or the CBI central computer system in which it can be embedded) is shown in Figure 2.

Note that provision has been made for the delivery of testing materials using paper-based or hard-copy materials, in place of the microfiche delivery medium, if such capability does not exist in the CBI system (shown by dotted flow-lines). The hardware components of the system are represented by shaded boxes.

Section II of this report addresses the hardware developed during Phase II, and Section III reviews the software development efforts of Phases II and III. A copy of the Users Manual is in Appendix B.



DOTTED FLOW LINES INDICATE POTENTIAL SYSTEM CAPABILITIES

FIGURE 2. BLOCK DIAGRAM OF PRIME ITEM CONFIGURATION

II. HARDWARE

For information concerning Phase I efforts toward the modification of the microfiche readers, consult the interim report by Kottenstette, et al. (1980).

Phase II efforts were principally concerned with the development of an interface between the grading microterminal and the Apple microcomputer. The serial interface module is designed to plug into the microterminal in the socket normally used for a memory module. Another socket on the adapter assembly is then available for plugging in the memory module. There is no change in the microterminal/memory module interface with the adapter assembly installed.

The module provides a serial interface port to an external device. All inputs and outputs to and from the external device are buffered. A baud rate of 110, 300, 600, 1200, or 2400 is selectable by moving a jumper wire on the printed ciruit card.

Data formatting and control are provided by a Motorola MC6850 asynchronous communications interface adapter (ACIA). The functional configuration of the ACIA is programmed by the software during system initialization. A crystal-controlled clock rate is provided by an MC14411 baud rate generator.

An on-board DC-to-DC converter provides plus and minus 12VDC for the MC1488 output line driver. This driver converts the ACIA data output in conformance with the specifications of EIA Standard No. RS-232C.

A 14-pin ribbon cable connector is provided for external input/output connections. A schematic of the serial interface is shown in Figure 3.

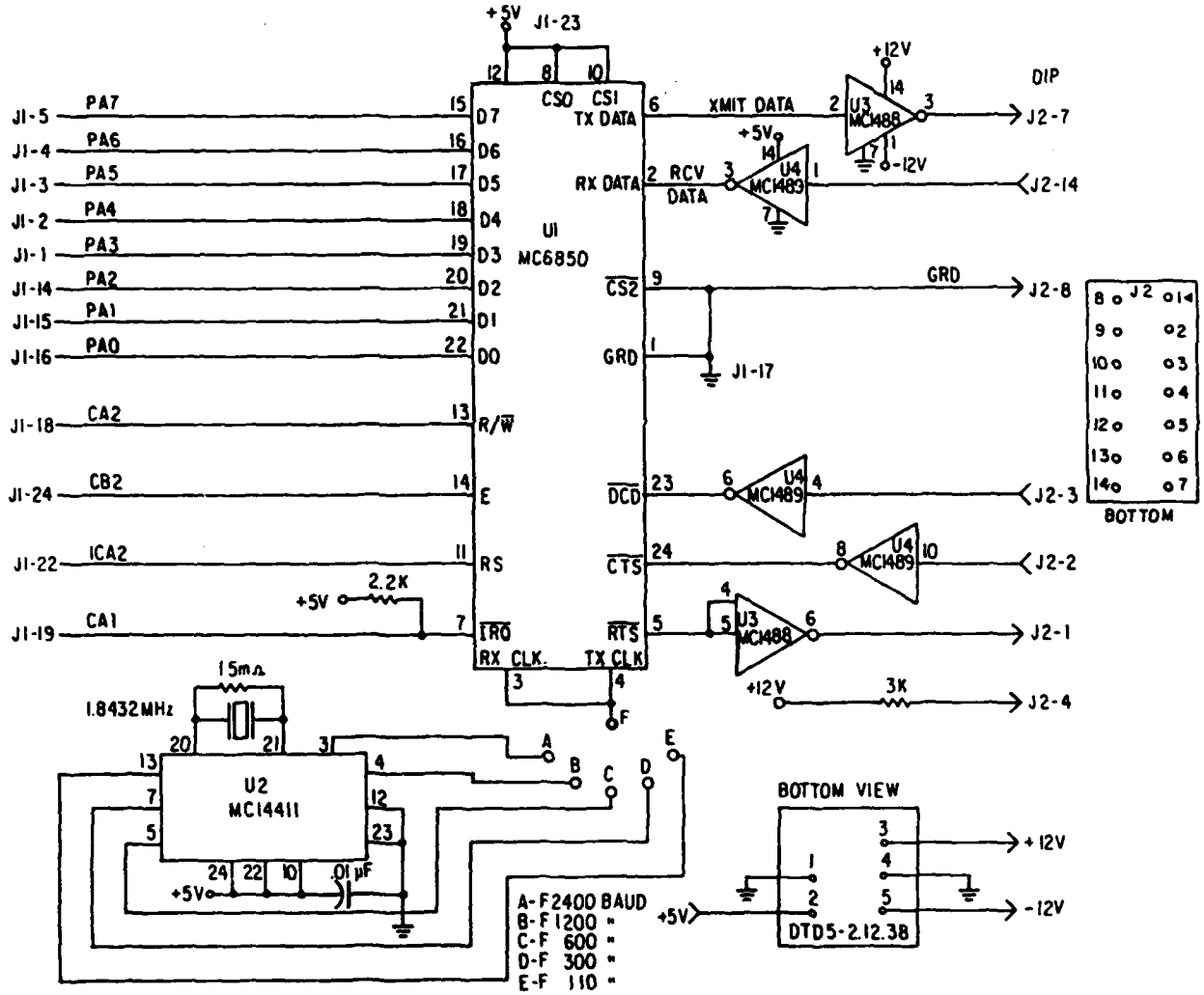


FIGURE 3. RS232 INTERFACE ACIA MODULE

III. SOFTWARE

A. PHASES I and II

During Phase I, the student test answer sheet had been replaced by the memory module. However, test booklets were still the delivery medium for examinations. Besides the low degree of test security afforded by paper exams, there was no way for instructors to administer test questions selectively. Selective testing capability can provide greater test security because examination questions are chosen from an item pool in a random manner, the selection dependent upon a pseudo-random number generator. Also, a student can be retested over failed objectives and be randomly assigned different questions over these same failed objectives. These capabilities can be taken advantage of if a test medium is used that can selectively present the questions the student is to answer, and one such medium is microfiche.

The use of microfiche under the control of the microterminal demonstrated in Phase I created the foundation for the work of Phases II and III. Microfiche, as a medium, is especially suited to interactive computer control because of its two-dimensional, row-by-column display format. Unlike text or linear film media, microfiche allows convenient access to any particular section of the fiche. This fits nicely with interactive-response, branching strategies that are usually associated with CAI. Additionally, since microfiche is a film medium, it provides high-resolution color graphics and text at a nominal cost. A desirable testing strategy in CBI is to have a pool of test items from which individual test items are selected for presentation under the control of an assignment algorithm. Usually, this selective-testing strategy requires a CRT-type CAI terminal and on-line access to the computer memory storing the test items. The objective in Phase II was to use the flexible storage capacity of microfiche and the control power of the microterminal to achieve a low-cost alternative for selective testing. The link between the microfiche reader and the student's micro-terminal had been successfully demonstrated at the conclusion of Phase I, and in Phase II the focus shifted to the interaction between the microterminal and a microcomputer, necessary for the implementation of selective-testing algorithms.

The software designed to provide the control necessary for microfiche-based testing was essentially developed in a two-part operation. First of all, a method was needed to generate the tape from which microfiche could be developed. This software package was labeled "EDITFICHE", and program documentation is included in Kottenstette, et al. (1980). This program aids authors in producing edited examination text by permitting (in conjunction with the AIS editor EDIT) relatively easy and quick text-formatting capabilities and, if desired, automated paper copy. Secondly, a method was needed that would: (a) provide control and coordination between the inquiry and response handling

capability of the student microterminal and the information (test) that had been formatted for display at the microfiche viewer, and (b) provide an interface between the grading microterminal and either the AIS or another comparable CBI system, such as a microcomputer-driven CMI control and testing capability. This second part of the software system is provided by two separate programs fully documented in a report entitled "MICROTERMINAL/MICROFICHE SYSTEM SOFTWARE PROGRAM DOCUMENTATION," (Steffen & Kargo, 1980).

EDITFICHE was developed in the CAMIL (computer assisted managed instructional language), and can only be used with the AIS program EDIT. Program EDIT is used to create a text file. EDITFICHE reads this file and allows a user to place the text within a frame. Paper copies or computer output microfiche (COM) tapes can then be produced.

Once a microfiche (or paper) test has been produced, another set of programs, collectively designated as the microterminal/microfiche/Apple system, is used to: (a) provide interaction between the student microterminal and the microfiche reader and (b) provide interaction between the instructor's grading microterminal and either AIS or a microcomputer-driven CMI system.

A program called FICHDSRC provides coordination between the student's microterminal and the microfiche reader. Depending upon the test booklet number that the student types into his microterminal, two types of exams can be administered. With one type of exam, the student can view the complete exam. The microfiche itself is divided into 3 parts, each part comprising a version of the test. A light on the microterminal indicates to the student which third of the microfiche will be lighted for use. This type of exam closely resembles a standard paper-and-pencil test because one entire exam version is available for the student's perusal at any time. With the second type of test, however, the student is permitted to view only selected questions from the entire microfiche. The student's identification number is used to produce an individualized "path" through the exam frames. The microterminal display indicates to which part of the microfiche the student is to position the reader's controls. If the student attempts to look at a question on the microfiche other than the one indicated in the microterminal display, the microfiche reader's lamp will turn off. Program FICHDSRC is resident within the microterminal firmware.

A set of three programs, written in pascal and stored in microcomputer diskette files, is used to provide interaction between the instructor's grading microterminal and the CMI system. With this software, the instructor can upload information from the student's memory module into the CMI computer's RAM, manipulate the data, and download data into the memory module. The instructor can determine what type of exam the student is to take and the testing strategy to be employed.

For detailed information concerning EDIFICHE and the microterminal/microfiche/apple system, consult the appropriate section ("Microfiche-based Testing") in Appendix C.

B. PHASE III

The three technological items (the microterminal, microfiche, and microcomputer) which have been integrated in a single configuration are sufficient to provide an immediate stand-alone testing facility to be used in Air Force resident technical training. An initial implementation of such a facility would best occur in the context of one of the courses used in the AIS project. A single room would be designated as a block test room, and would contain a microcomputer which would run the developed CMI/Test Administration and Evaluation software. Students would be registered, test keys would be created, tests would be allocated to blocks of instruction, and tests would be assigned and graded via the microcomputer-based program. Several individual student test stations, each containing a microterminal, would be established. Initially, tests could be presented in a paper format. The microcomputer, running any of several available word-processing software packages, could be used to develop and print new tests as frequently as desired. At this time, the test items generated via a word-processing capability would have to be manually matched to the generated answer keys produced by the Test Administration and Evaluation software. Students would respond to test items via the microterminal. Their answers would be stored in the memory module, which would be taken to the test administrator for grading by the microcomputer. All test records would be automatically updated. Automatic assignment of tests and the evaluation of tests and test items would be available. The test facility implementation just described could occur in either a conventional course or a self-paced course.

The microterminal/microcomputer system software package is designed with two user groups in mind: the student-trainee and the instructor/course designer. The system will allow the student to use a specially built microterminal and a memory module in lieu of test answer forms. Byproducts of this process are greater flexibility for the student in the learning environment and quicker feedback of evaluative information for the student. The system also provides a greater degree of flexibility for the instructor/course designer, who uses a grading microterminal interfaced to an Apple microcomputer for purposes of test management and course administration. These administrative and management tools will allow the instructor/course designer to:

1. Create and manage course/versions and blocks via the Course Editor Program (COURSEEDT)

2. Create and manage student group structures via the Group Editor Program (GROUPEDT)

3. Manage student data via:

   a. Group Editor (GROUPEDT) by displaying

      1) Names of groups (modifiable)
      2) Number of students in each group
      3) Data on students in a particular group

         a) student names
         b) student ID numbers
         c) course/version in which enrolled
         d) current block in which enrolled
         e) student score ratios, composed of a student's score divided by the mean score of that group (cannot be directly manipulated)

   b. Test Editor (TESTEDT) by displaying:

      1) Average student score on a test
      2) Standard deviation of average score
      3) Number of students in the average
      4) Number of times an answer was chosen by students on a particular test

   c. Course Editor (COURSEEDT) by displaying:

      1) Average score for students in a block
      2) Standard deviation of average score in a block
      3) Number of students in the average

   d. Student Editor (STUDENTEDT)

      1) Modify student name
      2) Modify student ID number
      3) Modify course/version name and number
      4) Modify group number name and number
      5) Modify the current block in which a student is enrolled
      6) Display student test information (& modify):

         a) most recent booklet number
         b) student's score for each block from Block 1 through the current block
         c) Contents of the past-test number array

4. Match exams with course/versions and blocks and create test parameters via the Test Editor (TESTEDT)

5. Assign tests to students and grade tests via Assign Test (ASSIGNTEST) and Grade Test (GRADETEST) programs

C. MAN/MACHINE INTERFACE

In order to develop software that was "user-friendly," consideration was given in the system design to "man/machine" interface techniques. The software programming procedures that were established would produce, as often as feasible, menu pages to which an operator could respond with a single, "active" key response. The display of data on the screen was formatted as consistently as possible. It was important to have the screen displays refreshed between the modifications performed on these data by the operator. The cursor is positioned for top-down data entry and writing. For the operator's convenience, a paged format, much like the leaves of a book, was preferred over scrolling techniques for displaying large amounts of data. To reduce operator confusion, the software consistently provides similar editing, aborting, and help functions across all file manipulations. Error messages are immediate and are given in plain English. The operator is provided with the capability for immediate error correction through direct cursor addressing and the fixed-page format. During computer processing, the operator is presented with appropriate messages in order to alleviate anxiety. Thus, operators are at all times aware that they are in control of the system. These interfacing techniques are embedded in all the applications programs within the system.

For a detailed description of system logic flow, see Appendix C (Programmers Manual).

IV. CONCLUSIONS AND RECOMMENDATIONS

The previous contract efforts and the three phases of the present effort have successfully resulted in the development and availability of an alternative CBI technology. Phase II efforts showed the feasibility of using the MT/MF system to enhance an existing CBI system by providing a low-cost input device (micro-terminal) for the collection of instructional information, such as student responses or course data, and an electronic storage and communication mechanism (memory module) which cost-effectively provides for the distribution of instructional data. This latter item, the memory module, is especially important because it permits the distribution of computer capabilities without necessitating major modifications to existing facilities for the installation of hardwire communication lines. Within a training facility, a student's legs become the communication system via the memory module. The microterminal, in the context of CMI, brings a set of interactive features associated with CAI to non-CRT-based instructional materials. Phase III efforts resulted in an integrated configuration with stand-alone capabilities independent of a CBI system such as the AIS.

With proper production facilities established, microfiche-based tests can be generated and then given via the MT/MF system. This would enhance the security of testing beyond that offered by printed test booklets. A particular advantage of microfiche is the ability to present high-resolution images with test items at a relatively low presentation cost. This possible configuration does not preclude the use or availability of CRT-based CAI, but, rather, it makes possible the consideration of a hierarchy of interactive instructional technology, providing a range of cost-effective options.

Finally, it is recommended that the MT/MF system technology be considered for support of CBI instruction beyond the category of testing. The microterminal is a generalized, objective-response-item input device. It can be used to interact with a variety of off-line instructional materials, for example, printed texts, film-strips, slides, videotape, and, of course, microfiche. As explained earlier in this report, microfiche is uniquely adaptable, among the film media, to the interactive response and sequencing characteristics of CAI. Additionally, with the recent availability of videodisc players containing a standard RS-232 computer interface, the microterminal is adaptable as an instructional control- and response-device for videodisc. Using the available technology and adding further work on integration, it would be possible to configure a self-contained learning center, using a microcomputer for management functions. Also, distributed microterminals could be used in individual work stations with a variety of presentation media to provide interactive instruction.

REFERENCES

1. Abramowitz, M., & Stegun I.A. (Ed)., Handbook of mathematical functions. National Bureau of Standards Applied Mathematics Series - 55 (Issued June, 1964), pp. 949-950.

2. Hull, T.E., & Dobell, A.R., Random number generators. SIAM Rev. 4 (1062), pp. 230-254.

3. Kirby, P.J., & Gardner, E.M., Microcomputer controlled, interactive testing terminal development. AFHRL-TR-76-66, AD-A035 731. Lowry AFB, CO: Technical Training Division, Air Force Human Resources Laboratory, October 1976.

4. Kottenstette, J.P., Microfiche applications in an individualized, self-paced learning system. AFHRL-TR-79-6, AD-A069 445. Lowry AFB, CO: Technical Training Division, Air Force Human Resources Laboratory, May 1979.

5. Kottenstette, J.P., Steffen, D.A., & Lamos, J.P., Microterminal/microfiche system for computer-based instruction: Hardware and software development. AFHRL-TR-80-17, AD-A090 974. Lowry AFB, CO: Logistics and Technical Training Division, Air Force Human Resources Laboratory, October 1980.

6. Steffen, D.A., Gray, G.C., Wasmundt, K.C., & Lamos, J.P., Development of a low-cost, stand-alone microterminal for support of testing and instruction. AFHRL-TR-78-50, AD-A060 215. Lowry AFB, CO: Technical Training Division, Air Force Human Resources Laboratory, September 1978.

7. Steffen, D.A., & Kargo, D., Microterminal/microfiche system program documentation. Unpublished report. Denver Research Institute, University of Denver, February, 1980.

APPENDIX A: STUDENT MICROTERMINAL INSTRUCTIONS

1. Make sure that the microterminal is connected to the power and is turned on.

2. Plug the memory module into the slot on top of the microterminal.

3. A light will appear next to the "ENTER SSN" message. Type in your student identification number.

4. A light will appear next to the "PRESS ENTER OR CLEAR" message. If you have correctly entered your student identification number, press the ENT key.

5. A light will appear next to the "ENTER BOOKLET NO." prompt. Type in your booklet number. If you type it in incorrectly, a "NO" message will flash on the screen. If this occurs, simply press any key on the keyboard to make the flashing message disappear. Reenter the booklet number and then press the ENT key.

6. A light will appear next to the "TEST IN PROGRESS" prompt.

7. A number with a question mark next to it will appear in the small screen. This is the number of one of the questions on the examination. Enter your response to this question by pressing the appropriate key, and then press ENTER.

8. If you wish to skip a question and come back to it later, press the SKP key.

9. If you want to look up all of the skipped questions, press the RVW key; the computer will respond by showing the numbers of all unanswered questions as many times as the key is pressed.

10. If a mistake is made in an entry, press CLR and enter the correct response.

11. When all questions have been answered, a light will appear next to the "ALL QUESTIONS ANSWERED" message.

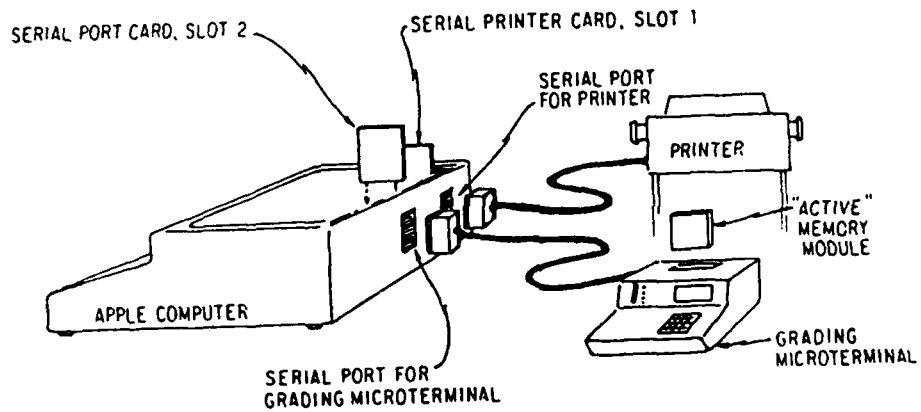
12. If you want to leave before finishing the examination, simply pull out the memory module. Whenever you plug it back in, the computer will pick up where you left off and will show you the question that was on the screen when the module was unplugged.

APPENDIX B: USERS MANUAL

HOW TO USE THIS MANUAL

This manual is intended to assist the instructor or course designer in using the microterminal/microfiche testing system. Section I shows how to connect the various hardware items of the system. Section II provides an overview of how the system works and offers some advice on what to do before using the system. Section III demonstrates how to actually use the system for setting up courses, assigning students to courses, and composing examination "structures" or formats that will be associated with the different courses/versions and blocks.

This manual was designed to be used as a tutorial for the first-time system operator and as a reference guide for the experienced user. The novice operator is strongly urged to read Section II carefully; this section advocates careful planning on paper (worksheets) to aid the designer in keeping track of the materials being created and organized.



FOR INSTRUCTIONS CONCERNING HOOKUP OF CRT AND DISK DRIVES, CONSULT APPLE MANUAL

SECTION I. MICROTERMINAL/MICROCOMPUTER/PRINTER CONNECTIONS

SECTION II

Before using the testing system, one should have a thorough understanding of the logic behind creating courses and tests, and enrolling students in courses. A few definitions are given below, to familiarize the user with some of the terms encountered in the course-planning processes.

COURSE: A course is composed of a series of blocks, and each block is represented by an examination. This examination covers the instructional material in the block (the material presented to the student via means other than the testing system). A course could be "PRECISION-MEASUREMENT EQUIPMENT (PME)", for example, and there could be up to 15 blocks or segments of instruction on the disk covering this topic. There is room on one disk for five courses.

GROUP: This term refers to a collection of students categorized for reasons convenient to the course designer. In traditional terms, a group could be called a class. However, no restrictions are placed on the course designer for a narrow definition of this term, and its use is optional. Because up to 300 students can simultaneously use the system, the designer may want some means of dividing this collection into smaller groups.

BLOCK: A block, as explained under COURSE above, is a stopping point within a course at which time an examination may be given to test a student's progress up to that point. In fact, several tests may be assigned to a block for purposes of selective testing. That is, if a student fails an examination for a given block, the instructor has the option of re-testing the student with a different test that covers the same material. The student may, in fact, rotate through this series of tests within a block any number of times, limited only by the student's endurance and the instructor's patience.

VERSION: Any course may be structured in different versions if the designer so wishes. In this way, courses of instruction and testing can be offered to different types of students. The testing material may be similar, but offered in different formats, depending on how the designer writes the tests.

TEST: The actual text of the examination will not be provided by the computer. What will be provided is how the test will be formatted (e.g., how many multiple-choice questions there are and how many choices there are per question) and the management portion of assigning and tracking both tests and students. The textual material may be provided to the student either on paper, via microfiche, or by any other means that the course designer elects.

One point to keep in mind is the logical flow of planning the curriculum testing. Figure B-1 provides a diagram of the system's functional flow. This order is particularly important the first time the system is set up. The Course Editor program establishes the groundwork of the system at its most elemental level. It is at this point that the overall structure of the system is mapped out, although it can be modified later. The Group Editor program must be executed prior to the Student Editor program for one simple reason: the designer knows who the students are but must decide how they are to be grouped, and a group framework or structure must be mapped out beforehand. Similarly, the Test Editor program must be used to help create the examinations that go with the different courses before they can be assigned to the students. After the system is set up, the instructor/designer can intervene as needed at these logic "stations" to modify or add testing materials and to enroll or remove students. Essentially, the "forms" must be present before they can be filled with the content.

A further aid to using the system is the worksheet. The easiest way to demonstrate its use is to actually create some courses and examinations, and then to assign students to those courses. By using worksheets, the designer need not be on-line to plan the curricula. The user can create worksheets based on the ones shown in this manual. Figure B-2, shows the first worksheet to use when starting from scratch. This Course Editor program worksheet will help to plan course/versions and the number of blocks within each course.

As can be seen in this example, four courses have been set up. The first course, "PRECISION-MEASUREMENT EQUIPMENT (PME)," is the only version of this course. It will be broken out into five blocks, or testing levels. There are two courses named "TRIAL COURSE," Version 1 and Version 2. They were assigned slightly different titles to help the designer remember that they cover similar materials. In the example, the number of blocks was chosen arbitrarily. In real application, these numbers would be dependent upon good course design.

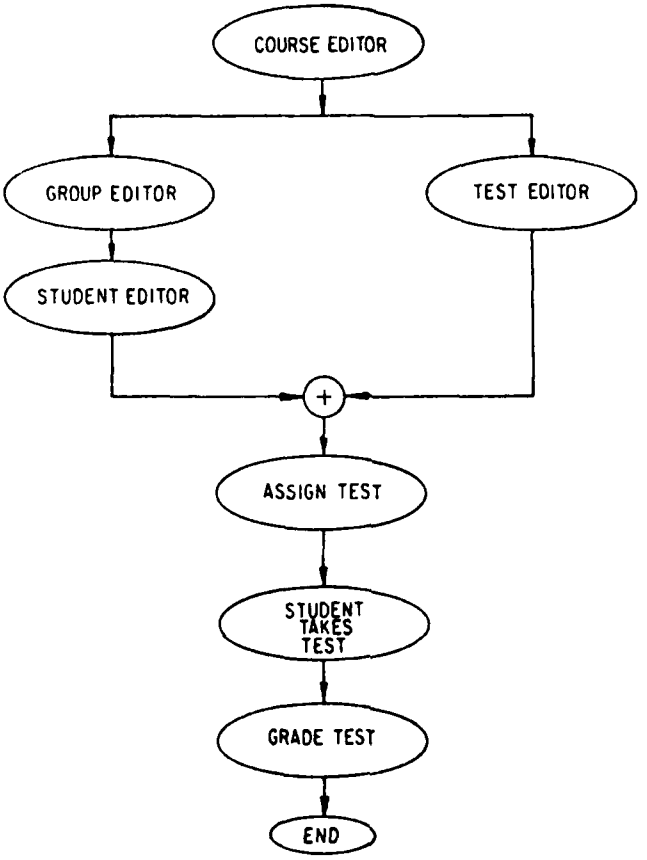


FIGURE B-1. SYSTEM SOFTWARE FLOW DIAGRAM

To create courses, versions of courses, and number of blocks per course

Course Number	Course Version Number	Number of Blocks	Course Name
1	1	5	PME
2	1	3	Trial Course
2	2	4	Trial Course #2
8	1	9	Toads and Warts

Figure B-2. WORKSHEET--COURSE EDITOR PROGRAM

After the courses have been established, the designer can either work on the Test Editor program portion of the design or go the other route, as was chosen here, to establish the groups to which students will be assigned. It is at this point that the designer must decide the group structure, or form. (See Figure B-3 for the Group Editor program worksheet.) In the examples given, it was decided to make the groups resemble classes--four of the groups even contain room numbers in their names. The designer can decide against groups, and, in this case, should establish one group--Group 0--and give it an appropriate name to designate its non-categorical structure. Later, the designer may distribute the students who are assigned to Group 0 into other groups. Notice that the Group Editor program does not assign students to the groups; it merely names groups into which students will next be placed through the use of the Student Editor program.

To use the Student Editor program Worksheet (see Figure B-4), the designer must assemble a list of the students to be enrolled. First, the columns under student identification number and name should be filled in, in whatever order is convenient to the designer. Then, by using the Course Editor program worksheet and the Group Editor program worksheet, which have already been filled in, the students should be assigned to the appropriate course/version, group and block.

Several things have been accomplished up to this point. The designer, who has established the courses, their versions, and how they are to be divided up into blocks, before using the worksheets, has now been able to quickly log this information into organized structures which are acceptable to the computer files and has assigned, or enrolled, students into those courses. Before proceeding to the next set of worksheets, the designer should decide how the students are to be tested and actually make out the examinations. However, the examinations do not need to be all worked out beforehand. In fact, the designer can write up as many examinations as are convenient and can record the information in the worksheets. The other examinations can be determined and formatted and then assigned to the appropriate courses at a later time.

The Test Editor program worksheet provides a convenient log of course/version and block assignments for the various examinations, as well as the various details about their formats. As can be seen, it is a good idea to have at least a rough draft of the examinations before commencing this portion of the process. This worksheet is divided into three sections (see Figure B-5). In this way, the novice system-user can become acquainted with the way the computer will present the displays, or menus, for the Test Editor program. Before the tests are formatted, the designer needs to assign the examinations to blocks and course/versions.

To create groups in which students can later be placed

Group Number	Group Name
0	General
1	TC Room A
2	TC Room B
3	TC Room C
4	TC Room D
5	Foxfire

Figure B-3. WORKSHEET--GROUP EDITOR PROGRAM

To enroll students in course/versions and blocks, and to assign students to groups

ID NUMBER	NAME	COURSE/VERSION [1]	GROUP # [2]	STARTING BLOCK # [3]
450987676	Wart Interbauer	8/1	5	1
000000001	George Schmultz	1/1	0	1
000000002	Jane Sharp	1/1	0	1
000000003	Tom Thumb	1/1	0	1
100000001	Gary Moore	2/1	1	1
100000002	Betty Crocker	2/1	1	1
110000003	John Kennedy	2/1	2	1
110000004	Beth Thompson	2/1	2	1

1. Look at the Course Editor program worksheet to find the Course/Version in which the student is to be enrolled.

2. Look at the Group Editor program worksheet to find the Group # to which the student is to be assigned.

3. Look at the Course Editor program worksheet to determine how many blocks are in each Course/Version. When assigning a new student, Block 1 is probably the desired level.

Figure B-4. WORKSHEET--STUDENT EDITOR PROGRAM

To make up numbers for tests, assign these tests to course(s)/ version(s) and block(s)

MENU PAGE 1		MENU PAGE 2		
BLOCK #	TEST #	COURSE # [1]	VERSION [2]	COURSE NAME [3]
1	1	1	1	PME
2	2	1	1	PME
1	6	2	1	Trial Course
2	7	2	1	Trial Course
1	6	2	2	Trial Course #2
2	7	2	2	Trial Course #2
1	3	8	1	Toads and Warts

MENU PAGE 3

Use with Part 2. Worksheet (Test Parameters)

1-3. Look at the Course Editor Worksheet to find the Course #, Version, and Course Name.

Figure B-5. PART 1. WORKSHEET--TEST EDITOR PROGRAM

To create the test parameters (format) after associating a test with particular course(s)/version(s) and block(s)

TEST #	PASSING SCORE (%)	STATUS [1]	NUMBER OF QUESTIONS	NUMBER OF CHOICES PER QUESTION
1	85	F	10	4
2	70	F	9	5
6	80	F	10	3
7	70	F	10	4
3	60	F	10	5

1. The Status Condition of the test is a future option that can be incorporated into the system. Because it is not operational at this time, a convenient letter to enter would be "F".

Figure B-5 (continued). PART 2. WORKSHEET--TEST EDITOR PROGRAM

Menu Page 1 requires the block assignments. Because the examinations must be identified by a number so they can be tracked, the designer must assign a number to each of the examinations that have been previously made up in rough form. Then, the designer must assign each of these examinations, by number, to a particular block. Although the menu doesn't ask which course/version this block is in, the designer must keep this in mind.

Menu Page 2 can be thought of as an extension to Menu Page 1. From left to right, the designer should fill in the course/version and course name for each of the tests. If the worksheet is read straight across, from left to right, all the information should line up, showing a test assigned to a particular block within a course.

Menu Page 3 is concerned with the actual formatting of the test. It is located in Part 2 of the worksheet, which has been placed on a separate page. The designer should write down the test numbers (which have just been assigned) in the left-hand column under Test #. Going to the next column, the designer should indicate what a passing score is for each examination. The status column should be ignored--at some time in the future this section may be used. For now, the designer should just notate an "F". In the next column, the designer should indicate how many questions are on each examination, and in the last column, the number of choices (multiple choice) there are for each question.

Again, the worksheet format for recording this information and the order in which it is to be recorded may seem awkward. However, this format is convenient when the data are transferred from the worksheet to the computer.

See Figure B-6 for the Assign Test worksheet. The bulk of this worksheet will not be filled out before using the computer. In fact, all that is necessary at this point is for the designer to write the student ID numbers in the left-hand column. When these numbers are entered into the computer at a later time, the computer will already know in which course(s) each of the students is enrolled. It will also know which examination has been assigned to each of the blocks in the course(s). The computer will match a student enrolled in a particular block to one of the tests assigned to that same block. From the computer display, the rest of the information--student name, course #, version, block, and test assigned (booklet number)--can be filled in on the worksheet, if desired.

The worksheets are handy tools. A designer who uses them before sitting down at the terminal can more easily concentrate on the computer displays.

STUDENT ID # [1]	STUDENT NAME	COURSE #	VERSION	BLOCK	TEST ASSIGN
450987676	Wart Interbauer	8	1	1	01030105015
110000003	John Kennedy	2	1	1	01063103000

1. Enter the Student ID #, and that student is randomly assigned a test from the particular course/version and block in which he is enrolled and to which a test has already been assigned.

Figure B-6. ASSIGN TEST WORKSHEET

SECTION III

This section of the manual will guide the user through the system software. It is written with the understanding that the software is self-documenting. That is, an operator, after starting to use the system, can become quickly oriented to its logic and can use this section of the manual as a quick reference guide.

Several system characteristics should be noted at this point. First, several steps are sometimes required to accomplish a given task. For example, a user who wishes to create a new course, must first use a special key (">" or "<") to move the cursor to a position marked "<not defined>". This insures that a course listed on the display will not be written over or modified. Next, the user must press another key ("B") so that a course can be created to eventually fill that slot on the display. The operator will then be "dropped through" to a deeper level (and a different display) that will permit creation of a course name and number.

This second characteristic (dropping through, or "trapdoors") is probably the most exciting facet of using this system. The user who is acquainted with computer games involving adventure mazes and the like will enjoy the way the software "cues" to provide the data required by the computer. Conversely, the software is designed to provide the course designer with a great deal of data that can be used in evaluating not only the students, but the course and the examinations themselves.

This data provision is another characteristic that should be touched on briefly. After the system has been in use long enough for the students to begin taking their tests, the Course Editor program can provide the designer with data that can be used in evaluating a course. The operator chooses a course and receives a printout showing statistical data on each of the blocks in that course: the average score on the examinations in that block, the standard deviation, the number of students used to determine the average, and the number of tests in that block. Test booklet numbers for each of the blocks are printed out immediately below this entry so the operator can associate the booklet numbers (tests) with each block.

The Test Editor program, for another example, provides a printout called an "item tally" for purposes of item analysis. A grid is produced, with the question numbers composing the left vertical axis, and the choices in the question (A, B, C, etc.) providing the top horizontal axis. The grid marks the correct answer positions with an asterisk. Also printed is a number in each position, showing how many times the students selected a particular answer. Therefore, if the second question, for example, has a correct answer of "D", but only three students have ever chosen that answer while the majority of the students have chosen answer "B", the instructor can take appropriate

action by altering either the test question, the answers, or the course instructional material.

It is not the purpose of this manual, however, to dwell on the details of the software. The user should have the opportunity to become familiar with the more general functions of the system and should be allowed to explore independently thereafter.

SYSTEM INITIATION

1. Place the Apple 1 disk into Drive 1 and the Data disk into Drive 2.

2. Turn on the power. (The switch is on the rear panel of the Apple computer.)

3. The following information should appear on the screen:

```
COMMAND: E(DIT, R(UN, F(ILE, C(OMP, L(IN

WELCOME APPLE 1, TO APPLE II PASCAL 1.1
BASED ON USCD PASCAL 11.1
CURRENT DATE IS
```

This information means that the computer is in the Pascal operating system. This system is used to call up the various programs that will enable the user to manipulate student, course, and test information. To tell the system to execute (or run) a particular program, first type in: X. The system will respond with: "EXECUTE WHAT FILE?" This means "Which program should be run?" There are six programs, each with its own title:

FUNCTION DESIRED	PROGRAM NAME
COURSE EDITOR--Establish Courses	COURSEEDT
GROUP EDITOR--Establish Groups	GROUPEDT
STUDENT EDITOR--Assign Students to Groups	STUDENTEDT
TEST EDITOR--Match Tests with Courses and Create Parameters	TESTEDT
ASSIGN TEST--Assign Tests to a Student	ASSIGNTEST
GRADE TEST--Score Tests	GRADETEST

These functions are listed in the order necessary for starting the system from scratch (no courses, versions, tests, students, or groups). Once the system has been in operation, they can be accessed in any order required.

After the prompt "EXECUTE WHAT FILE?" appears, simply type in the name of the desired program exactly as it appears above, under Program Name; then, press the RETURN key.

COURSEEDT

In the upper part of the display are listed the courses, their versions, and the course titles or descriptions. The title "<not defined>" indicates an empty slot in which a course can be created. As can be seen, up to five courses can be established on each Data Disk. If there are five "<not defined>" statements, no courses have yet been set up.

In the lower part of the screen are the editing tools. Of special importance are the two symbols at the bottom of this list: ">" = move cursor down, and "<" = move cursor up. The cursor is the square white box sitting to the left of one of the courses listed at the top of the screen. It acts as a pointer. By pressing down on the SHIFT key and, at the same time, pressing either the ">" key or the "<" key, the cursor can be made to point to any of the courses. It is important that the operator become familiar with this function. As an exercise, move the cursor up and down the list a few times.

When the cursor is pointed at the appropriate "slot," the editing keys (A, B, C, or D) can be pressed to perform the listed function. For example, if a course needs to be created, move the cursor to any of the slots marked "<not defined>". Then, press the "B" key. The program will guide the user from that point. Be sure that a Course Editor program worksheet is handy to facilitate data entry.

If it is desired to inspect or to make some changes in a course that has been previously set up, move the cursor to that course's name and press the "A" key. As usual, the computer will provide assistance from that point.

To exit from this program and to to get back to the Pascal operating system, press the "Q" key. When back in the operating system, pressing the "X" key and responding to the computer message with another program name will place the operator in another program.

Note: When back in the operating system, and no more programs are desired, simply remove the disks from their drives and turn off the computer.

GROUPEDT

When this program is executed, four menu items appear on the screen, allowing the operator to see information concerning groups that have been previously formed, or for creating a group. Some definitions are needed here to allow the operator to choose what is desired.

GROUP SUMMARY: A Group Summary is simply a listing of the different groups (if any) that exist. The Summary does not show who the students are in a group: it only shows how many students are in that group.

An operator who wants to see who is in a given group should press the "C" key, which will allow an inspection or modification of a given group. Pressing this key will also initiate the process for creating a group.

When the "C" key is pressed, a prompt will appear, asking the operator for the group number that is to be examined. If a group number is entered that was not previously entered into the computer, the operator will be allowed to create a group under that number. If the group number has been previously entered, the computer will display the information it has stored in its memory concerning the requested group.

GROUP ROSTER: If the operator decides to inspect a group that exists, the computer will ask if a group roster is wanted. A Group Roster contains a listing of the students in that particular group. (The names will appear in an abbreviated form so all the data can appear on the screen.) Other student information, such as a score ratio on the examinations taken up to that point, will also appear.

Note: If a key is pressed to enter data and nothing happens, press the **RETURN** key. This will push the information into the computer.

STUDENTEDT

When the display appears, at the top can be seen the message "**STUDENT ID#**", followed by a group of zeros. The cursor will be resting on the first digit.

1. When a Student ID # is typed in by the operator and the computer has it on file, the process is initiated for inspecting and/or modifying that student's data.

2. If the computer does not have the typed-in Student ID # on file, the process is initiated for entering student data. At the bottom of the display, a menu will appear. An operator who wishes to enter a new student's name and other information will press the "B" key and follow the computer's guidance from that point. The student can now be enrolled in a course/version, group, and block that already exist. If the course does not exist, the operator must exit the program and call up the COURSEEDT program. If the group does not exist, the GROUPEDT program must be called up.

Note: Always pay careful attention to the editing tools listed at the bottom of each menu. They vary from task to task.

TESTEDT

In the upper left-hand corner of the display, underneath the TEST EDITOR title, there appears a page number. These page numbers correspond to the Menu Page numbers on the Test Editor program worksheet.

On page 1, the data that should be entered are the block number to which a test will be assigned (keep the course/version in mind) and the test number that the user wishes to use to designate that particular test. Remember that a course/version has only so many blocks, the number of blocks that was allocated for it when the course was set up under the COURSEEDT program. Refer to the Course Editor program worksheet, when necessary, to help keep track of what information is being entered into the computer.

Of course, if the block number and test number that are entered are already in the computer's memory, the computer will say so, and will allow some of the test data to be modified. Otherwise, the test will have to be created.

Remember that the cursor is a very important tool. Thus, on Page 2, if a new test is being created, be sure to move the cursor to the appropriate course/version with which the test is to be matched or associated, and then press the appropriate key. On Page 3, the actual parameters of the test can be mapped out.

When the user has finished mapping out the parameters, an important display will appear, labeled as Page 4. This display shows how the computer has randomly ordered the correct answer sequence. If the order looks like a student could detect a right-answer response according to a pattern, have the computer increment or decrement the 'randomizer' or number, used by the computer to calculate the answer array by pressing either the ">" key or the "<" key.

When satisfied with the sequence of correct answers, go back to Menu Page 3 to see if any other modifications are needed. If there are none to be made, be sure to press the "D" key so the computer will save the set of parameters for that examination.

As an exercise, when back on Menu Page 1, press the "D" key (allow time for the computer to adjust), and Page 5 will appear. Press the "A" key (or the ">" key if no printout is desired) to examine the item tallies (previously commented upon in the introduction to this section of the manual). If there are several tests already in the system, choose an old one (by block and test number) and see if anyone has taken the test to date.

ASSIGNTEST

When the display first appears, the operator is asked to type in a Student ID #. When that has been entered, the computer will search its files to locate the course and block in which the student is currently enrolled. If the test(s) for that block have already been created, a display listing the examination(s) for that block will appear.

The computer will randomly select an examination from that block which has not been assigned and will place it at the top of the list. If the operator agrees that that examination should be the next one assigned, the appropriate key should be pressed. Otherwise, the instructor may choose a more appropriate examination. Also shown on the list are tests failed by or previously assigned to that student.

The order in which exams are listed are:

Position 1: One of the following:

a. a test randomly chosen from the tests not taken by the student;

b. a currently-assigned test which has not been taken;

c. the oldest, graded test from all the exams if the student has completed all the exams and they have been graded. If desired, the instructor may reassign one of these exams.

Position 2: Tests which have not been taken (besides the one in Position 1), if any.

Position 3: Tests which have been taken, if any. The word "TAKEN" appears after the booklet number.

GRADETEST

To use this program, several procedures must be followed. First, take the student's memory module and plug it into the grading micro-terminal that is connected to the computer. Next, execute the GRADETEST program.

If the memory module contains an incomplete test, a "VALUE RANGE ERROR" message will appear. The instructor either can instruct the student to complete the examination or can reinitialize the module.

If the test was completed, the computer will score the examination, save the results, and provide the information to the instructor via a display. Once a test has been graded, the software will automatically reinitialize the module for further use.

APPENDIX C: PROGRAMMERS MANUAL

INTRODUCTION

The microterminal/microcomputer system software package, written in UCSD Pascal, is designed with two user groups in mind: the student-trainee and the instructor/course designer. The system will allow the student to use a specially built microterminal and a memory module in lieu of test answer forms. Byproducts of this process are greater flexibility for the student in the learning environment and quicker feedback of evaluative information for the student. The system also provides a greater degree of flexibility for the instructor/course designer, who uses a grading microterminal interfaced to an Apple microcomputer for purposes of test management and course administration. These administrative and management tools will allow the instructor/course designer to:

1. create and manage course/versions and their sub-parts or blocks;
2. create and manage student group structures;
3. manage student data;
4. associate or match examinations with course/versions and blocks, and to create basic test parameters;
5. assign tests to students and to grade the tests;
6. provide statistical information on tests and courses, to aid in their evaluation.

MAN/MACHINE INTERFACE

In order to develop software that was "user-friendly," consideration was given in the system design to "man/machine" interface techniques. Software programming procedures were established which would produce, as often as feasible, menu pages to which an operator could respond with a single, "active" key response. The display of data on the screen was formatted as consistently as possible. It was important to have the screen displays refreshed between the modifications performed on these data by the operator. The cursor is positioned for top-down data entry and writing. For the operator's convenience, a paged format, much like the leaves of a book, was preferred over scrolling techniques for displaying large amounts of data. To reduce operator confusion, the software consistently provides similar editing, aborting, and help functions across all file manipulations. Error messages are immediate and are given in plain English. The operator is

provided with the capability for immediate error correction through direct cursor addressing and the fixed-page format. During computer processing, the operator is presented with appropriate messages in order to alleviate anxiety. Thus, operators are at all times aware that they are in control of the system. These interfacing techniques are embedded in all the applications programs within the system.

To facilitate "user-friendly" data entry techniques, the following procedures are used for operator input. Procedure GET_STRING (X,Y OLD_STRING, MIN_CHAR, MAX_CHAR, NEW_LENGTH, MAX_LENGTH) insures a limitation on string input that would not confuse the operator. The GET_IN and GET_REAL procedures allowed input of integers and real numbers, respectively.

For an example of a five-section menu-page procedure, see Figure C-1, a flow chart demonstrating data entry. After program initialization, the cursor is positioned in a consistent location for start-up. The display presents a fresh view of data whenever the data are modified.

There are three types of data entry possible: string input, which is immediately processed; true/false or yes/no data input with a single key-stroke; and integer input. A pointer adjustment allows the cursor to be advanced to the appropriate location for subsequent data entry (e.g., modification of a data display). The processing part of the interface is composed of actual computation.

Figure C-2 shows an example of the procedures followed for a pure menu display with no data entry. The procedures for data entry and pointer adjustment are replaced with a call to GET_CHAR (for branching). Screen refreshing is mandatory due to the replacement of the menu with the display from the procedure called.

SYSTEM SOFTWARE

A functional block diagram of system software is shown in Figure C-3. Because files must be constructed before they can ever receive data, the system flow will be described from the perspective of the operator. This section of the manual will describe the programs in the order in which they must be executed to establish various course/versions, groups, and student files and test files:

1. Course Editor/Course File (COURSEEDT)

2. Group Editor/Group File (GROUPEDT)

3. Student Editor/Student Files (STUDENTEDT)

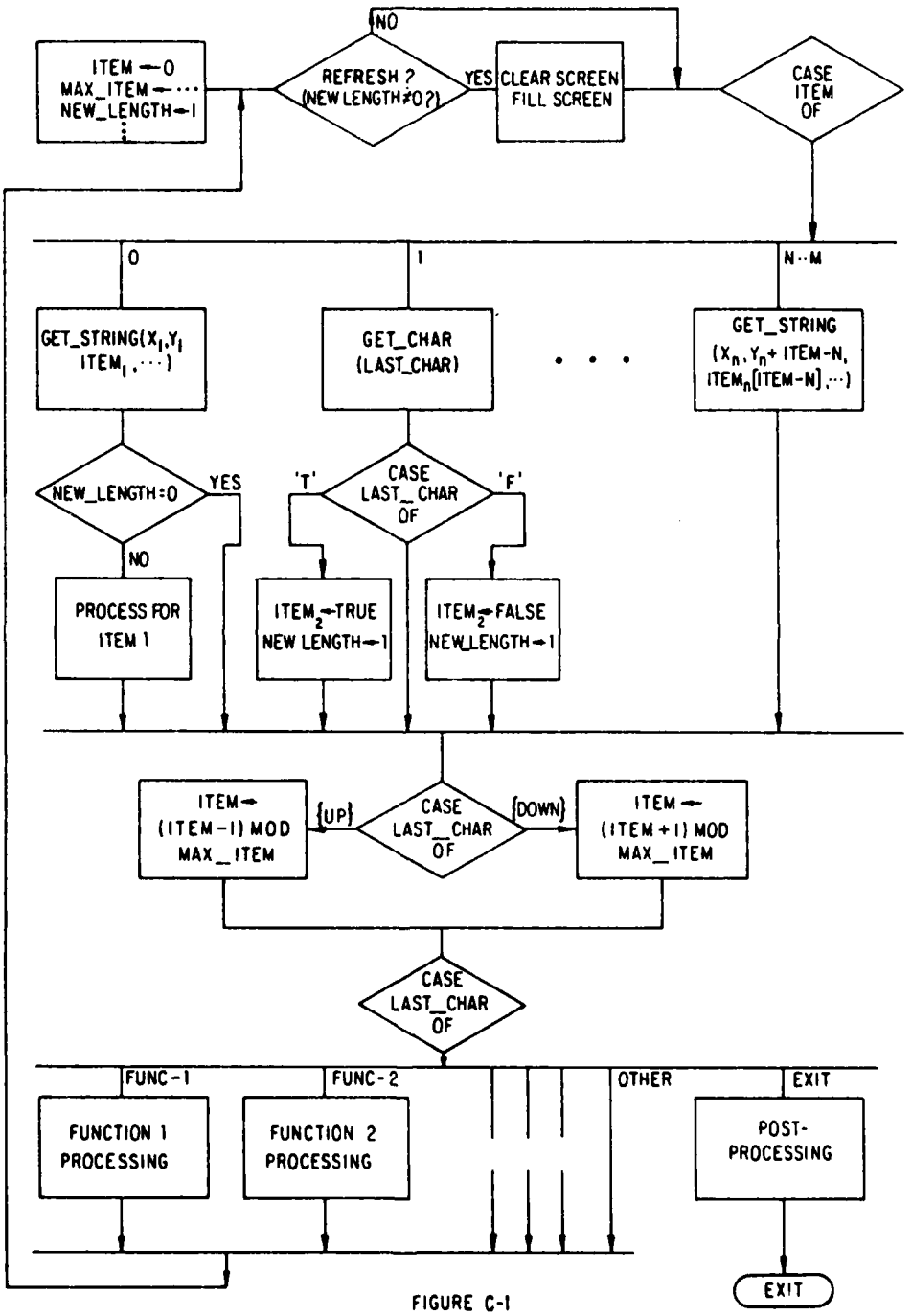


FIGURE C-1
FLOW CHART FOR APPLICATION PROGRAM. MENU PAGE WITH DATE ENTRY

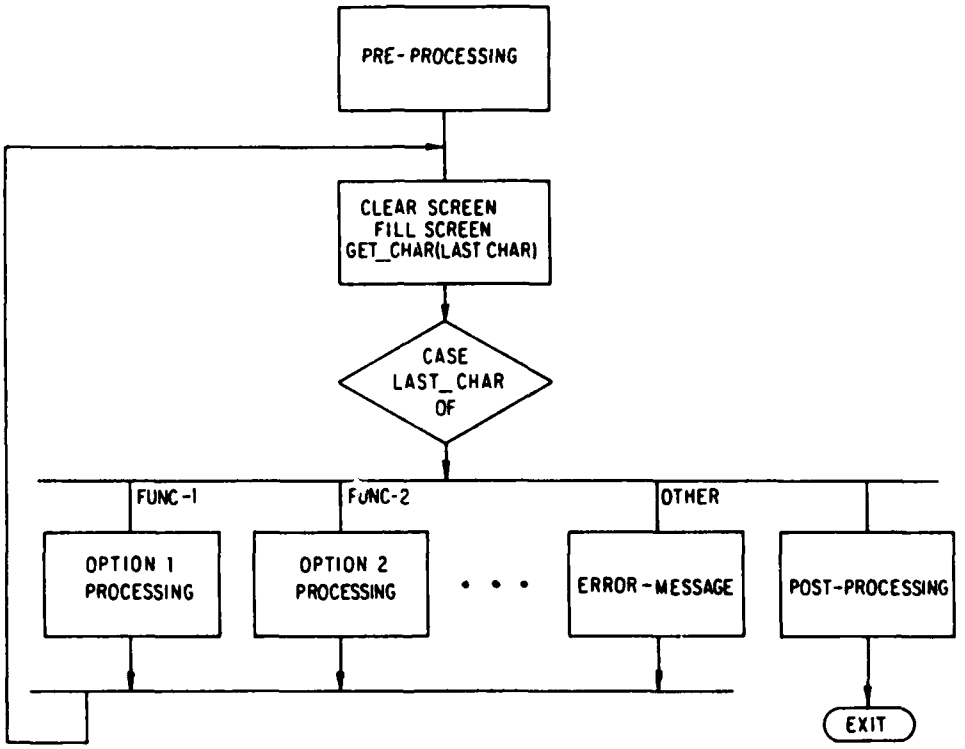


FIGURE C-2
FLOWCHART FOR MENU PAGE (OPTION SELECTION ONLY)

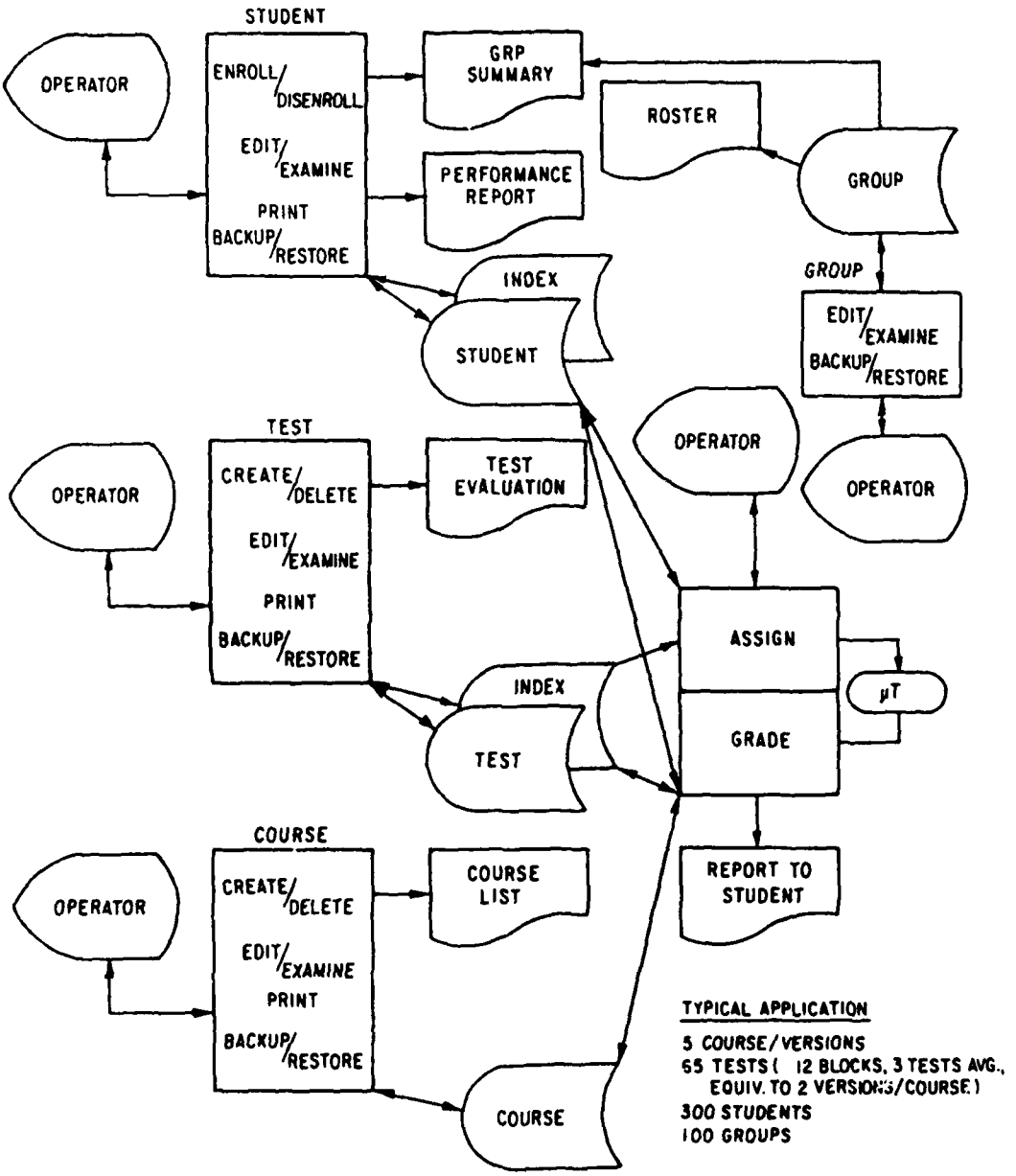


FIGURE C-3
FUNCTIONAL BLOCK DIAGRAM

4. Test Editor/Test Files (TESTEDT)

5. Assign Test (ASSIGNTEST)

6. Microterminal Software (EROM "Firmware"), used by the student

7. Test Grading Program (GRADETEST)

All program interfaces are via the Index files and/or the Data files.

COURSE EDITOR (COURSEEDT)

The Course Editor program allows the system manager to manipulate the data contained in the Course File. The functions of this program are:

1. to inspect/modify course data;

2. to create a course;

3. to print course data;

4. to delete a course.

The files which are used and/or accessed by the Course Editor are:

1. the Course File;

2. the Student Index File, used when:

   a. modifying course data;

   b. checking the procedural validity for deleting a course.

3. the Test Index File, used when:

   a. modifying course data;

   b. finding the number of tests within a block.

4. the Test Data File, used when displaying booklet numbers of tests for a block.

The initial use for this program is to create the Course File. The limits on the number of digits and characters allowed for operator data input are:

1. Course Number: 1-999;
2. Version Number: 1-99;
3. Block Number: 1-15;
4. Course Name: 30 characters (alpha).

The following information can be obtained by calling up this program:

1. Course Name/Number/Version;
2. the number of blocks in a given course/version;
3. the average score for students in a block;
4. the standard deviation of the average scores for each block;
5. the number of students in the average;
6. the number of tests (from the Test Index File);
7. the test booklet numbers (from the Test Index File/Test Data File).

A flow diagram for the Course Editor program is shown in Figure C-4.

GROUP EDITOR (GROUPEDT)

The Group Editor program allows the system manager to manipulate the data contained in the Group File. The functions of this program are:

1. to display/print the Group Summary;
2. to edit/examine the Group File;
3. to print/display the Group Roster;
4. to edit the Group Name (create, modify, delete).

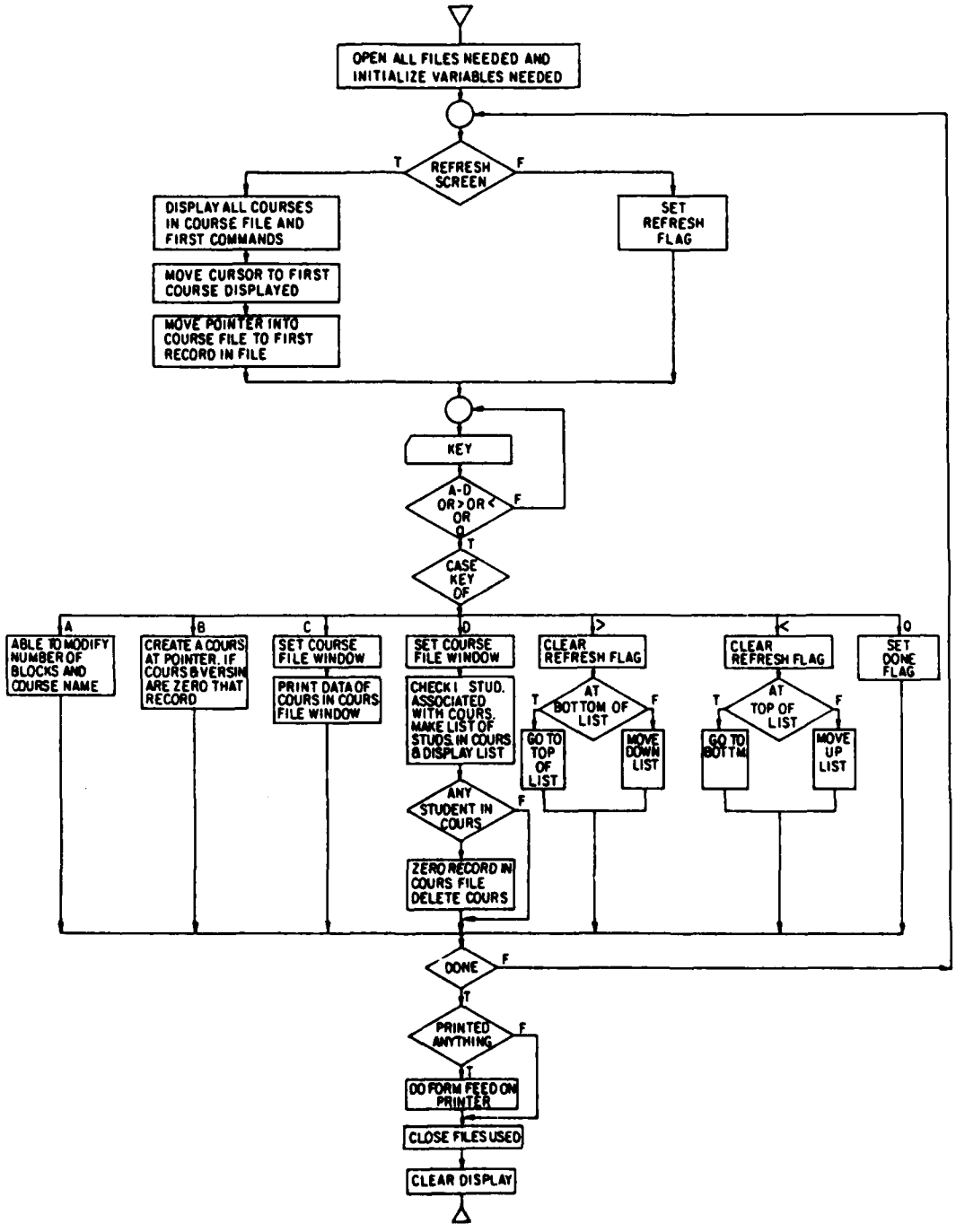


FIGURE C-4. COURSEEDT

The files which are used and/or accessed by the Group Editor are:

1. the Group File, for the Group Summary;

2. the Student Index File, for the Group Roster and Summary;

3. the Course File, for the Group Roster.

The initial use for this program is to create the Group File and the groups. The limits on the number of digits and characters allowed for operator data input are:

1. Group Number: 0-99;

2. Group Name: 50 characters (alpha).

The following information can be obtained by calling up this program:

1. the Group Summary, which contains:

    a. groups that exist by number and name (not named "EMPTY");

    b. the number of students in a group;

    c. the number of students in the system.

2. the Group Roster, which contains:

    a. the Group Number;

    b. the number of students in a group;

    c. the Group Name.

3. student data:

    a. Student Name;

    b. Student Identification Number;

    c. the Course/Version Number(s) in which the student is enrolled;

    d. the current block in which the student is enrolled;

    e. Student Score Ratio.

4. data on all courses/versions in the system:

   a. the Course/Version Number;

   b. the Course/Version Name.

A flow diagram for the Group Editor program is shown in Figure C-

STUDENT EDITOR (STUDENTEDT)

The Student Editor program allows the operator to manipulate the data contained in the Student Files. The functions of this program are:

1. Display #1: student selection--displays a student identification number. The operator edits student identification number by typing over the currently displayed number if a different one is desired. If the identification number is not on file, a message will cue the operator for the entry of a new student record into the system. The program then calls up Display #2, which allows the operator to enter student data.

2. Print student's data--prints all of the data in Display #2 and Display #3.

3. Delete this record--warning query flashed to ascertain operator sincerity.

4. Advance or retreat to following or previous student record in the system.

5. Proceed to Display #2.

6. Display #2: general student information--displays student identification numbers, student name, course/version number(s) and name(s), group number and name, the current block in which the student is enrolled, and the student's score ratio. All data, except the score ratio, may be modified. The score ratio computation is changed whenever the block number or block score is changed. The score ratio is equal to the sum of the student's scores from Block 1 through the current block, divided by the sum of the average scores from the Course Files (Block 1 through the current block). If the current block number is changed by the program (i.e., when a student graduates from one block to the next), the past-test number array is zeroed.

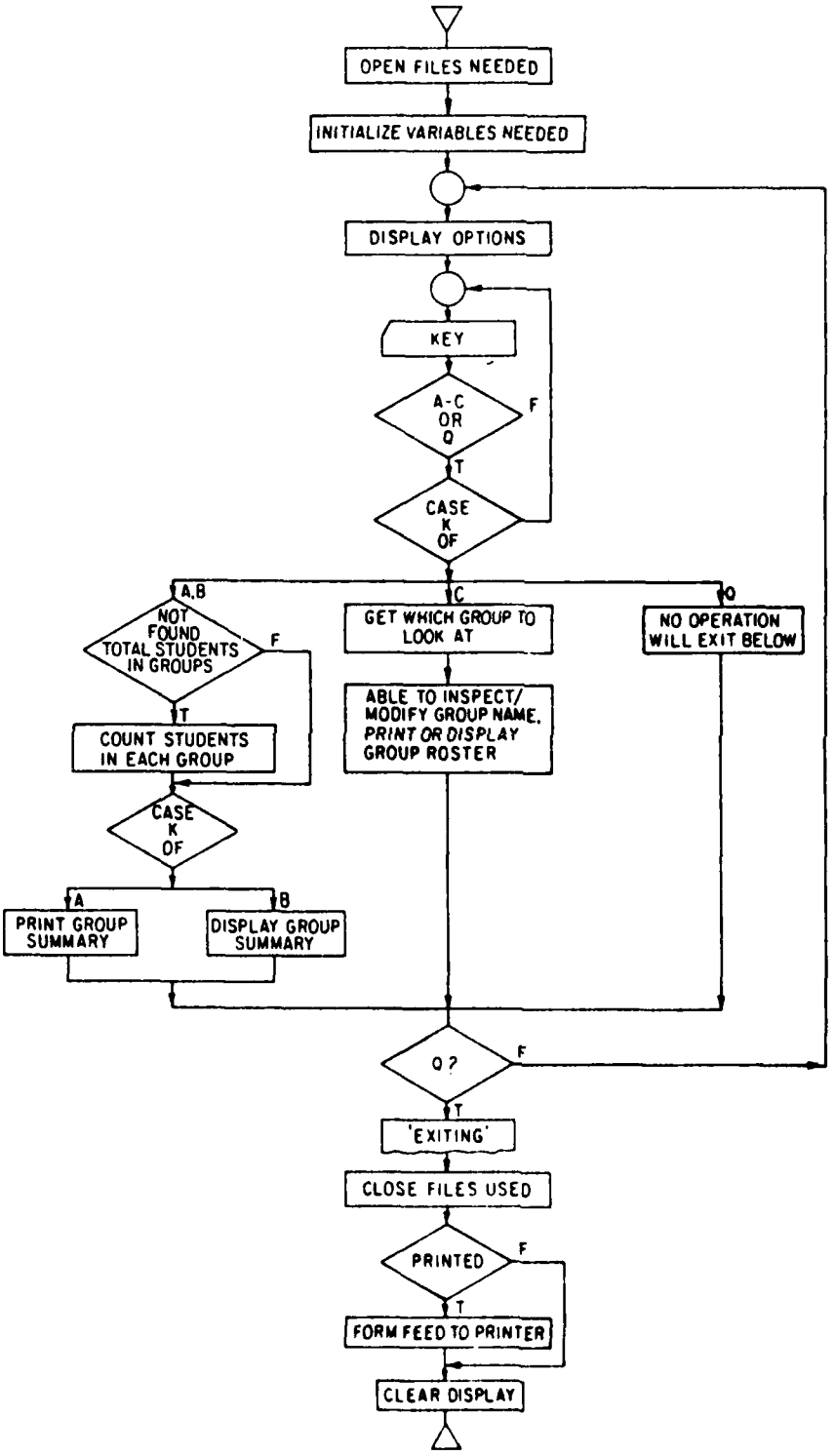


FIGURE C-5. GROUPEDT

7. Display #3: student test information--displays the most recent booklet number, the score for each block from Block 1 through the current block, and the contents of the past-test number array. The operator is allowed to modify scores, an action which will affect the score ratio.

The files which are used and/or accessed by the Student Editor program are:

1. the Student Index File and the Student Data File;

2. the Course File, when displaying a Course Name and when checking for a valid Course/Version;

3. the Group File, when displaying a Group Name and when checking for a valid Group Number.

The initial use for this program is to create student records by creating the Student Index File and the Student Data File. The program records student names and identification numbers, and assigns them to a course/version, a group, and a block. The limits to the number of digits and characters allowed for operator data input are:

1. Student Identification Number: 11 digits;

2. Student Name: 30 characters (alpha).

The information available by calling up this program has been detailed above in items 1, 2, 6, and 7 in the paragraph on the Student Editor program functions.

A flow diagram for the Student Editor program is shown in Figure C-6.

TEST EDITOR (TESTEDT)

The Test Editor program allows the operator to create, modify, or delete tests within the system. Note that within this system, a test exists only as a Booklet Number, a Pass/Fail Score, and some historical records of its use. The Booklet Number is composed of the following information:

b1--the most significant digit of the block number;

b2--the least significant digit of the block number;

t1--the most significant digit of the test number;

t2--the least significant digit of the test number;

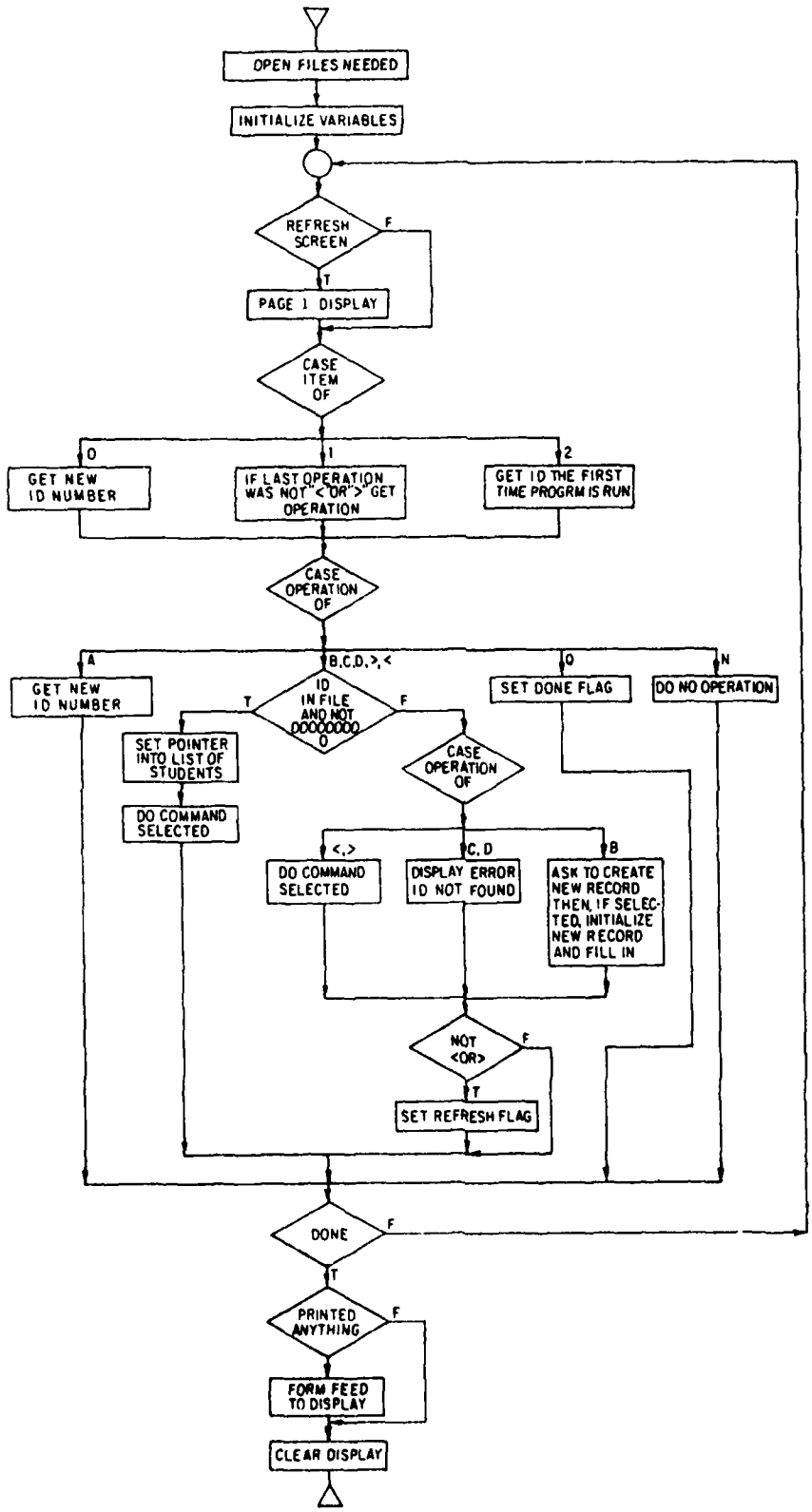


FIGURE C-6. STUDENTEDT

ck1--the check digit for the first data group;

n1--the next-to-the-least significant digit of the number of questions on the test;

n2--the least significant digit of the number of questions on the test;

q--the number of items on each test question;

v1--the most significant digit of the test randomizer;

v2--the least significant digit of the test randomizer;

ck2--the check digit for the second data group.

The text for the test (instructions, stem, answers, and distractors) must be maintained separately on paper or some other medium. The functions of the Test Editor program are:

1. Menu Page 1: test identification. This part of the program allows the operator to input the block number and test number of the test to be viewed. These data are used by the program to compose the first four digits of the booklet number. The Test Index File is searched for an association between these test data and a course/version. If the information is located, the display conveys this information to the operator and the program computes the appropriate digits for the booklet number. If the block number and test number entered as data are not found as a pair in the Test Index File, the operator has the choice of creating the test or editing the block number and test number just entered. Before a test can be created, the operator must associate the examination with a course/version, and then proceed to the test-creation process (Menu Page 2). If a test-course/version association has previously been established, the operator may break this match by removing the association or by modifying it. If all association is dissolved, the test record is removed, thereby destroying test history information.

2. Menu Page 2: test specification/modification. This portion of the program allows the operator to create new tests or to modify existing ones. Test data which can be created or revised are:

   a. the number of questions on the test;

   b. the number of items or choices per question;

c. the test randomizer (see Appendix D for the algorithm).

d. the status of the test. This is a possible future enhancement which could provide for:

i. Active test with individual student records being saved in the Test History File;

ii. active test with individual student records not being retained in the Test History File;

iii. inactive test, in the process of being created;

iv. inactive test, in the process of being deleted.

e. the date of the status assignment or modification;

f. the pass/fail score percentage.

Modifications to the status, status date, and passing score will alter corresponding data in the Test File. Modifications to the number of questions on the test, the number of items per question, and the test randomizer will result in the following to be performed by the program:

a. check digit computation;

b. a new Booklet Number display;

c. a check on Booklet Numbers to be run.

The operator may then print the results of the modifications, return to Menu Page 1 without changing the Test File, remodify the test parameters if the changes are unsatisfactory, or change the Test File to reflect a new booklet number, which will zero out the test score and item tally.

3. Menu Page 3: test evaluation. This portion of the program displays or prints out an item tally, a listing of the answers to the questions on the test, with the correct answers marked with an asterisk, and the number of times a particular answer on the test has been selected by the students. This information allows the course designer to verify test difficulty (percentage correct/test average) and discrimination (percentage correct/total). A printout or display of the answer array calculated by the randomizer is also available.

The files which are used and/or accessed by the Test Editor are the Test Index File and the Test Data File.

The initial use for this program is to create tests and to assign them to course/versions and blocks. The limits on the number of digits and characters allowed for operator data input are:

1. number of questions per test: 1-50;

2. number of items per question: 1-5;

3. test number: 1-99.

The following information can be obtained by calling up this program:

1. from the Index File:

   a. the course/version in which the test is located;

   b. the test number (key for Index File);

   c. test identification (pointer to Test File);

   d. the block number (key for Index File);

   e. the number of questions;

   f. the number of items per question;

   g. check digit 1;

   h. check digit 2;

   i. the randomizer;

   j. status check;

   k. passing score;

   l. the average score on the test;

   m. the standard deviation of the average score;

   n. the number of students used to calculate the average;

   o. the number of times a particular answer was chosen on the test (item tally);

   p. the answer array.

See Appendix D for the mathematics involved in the randomizer portion of the program, and the relationship between test descriptions, test booklet numbers, and correct answer arrays.

A flow diagram for the Test Editor program is shown in Figure C-7.

ASSIGN TEST PROGRAM (ASSIGNTEST)

The Assign Test program allows the operator to assign a test to a student. The program will choose on a random basis an available test to be administered (with the option for operator override) and will provide adequate assignment information. The program will allow the operator to enter the student identification number. The Student File is read to determine the course/version and block in which the student is currently enrolled. The Test Files are read, and a display depicting all of the tests which are available for that course/version and block is presented to the operator. The positions of the different examinations, from the top of the list to the bottom, are:

Position 1: One of the following:

a. a test randomly chosen from the tests not taken by the student;

b. a currently assigned test which has not been taken;

c. the oldest, graded test from the past-test number array if all of the examinations in a block have been taken. (This permits recycling through block examinations.)

Position 2: Tests which have not been taken, if any.

Position 3: Tests which have been taken, if any. These are indicated by the word "TAKEN" after the booklet number.

The instuctor is cued to assign a randomly selected test or to override this function to choose another test available for that course/version and block. The program will update the past-test array, if more than five entries have been made, by leaving the last entry identifiable and deleting the oldest entry.

The files which are used and/or accessed by the Assign Test program are:

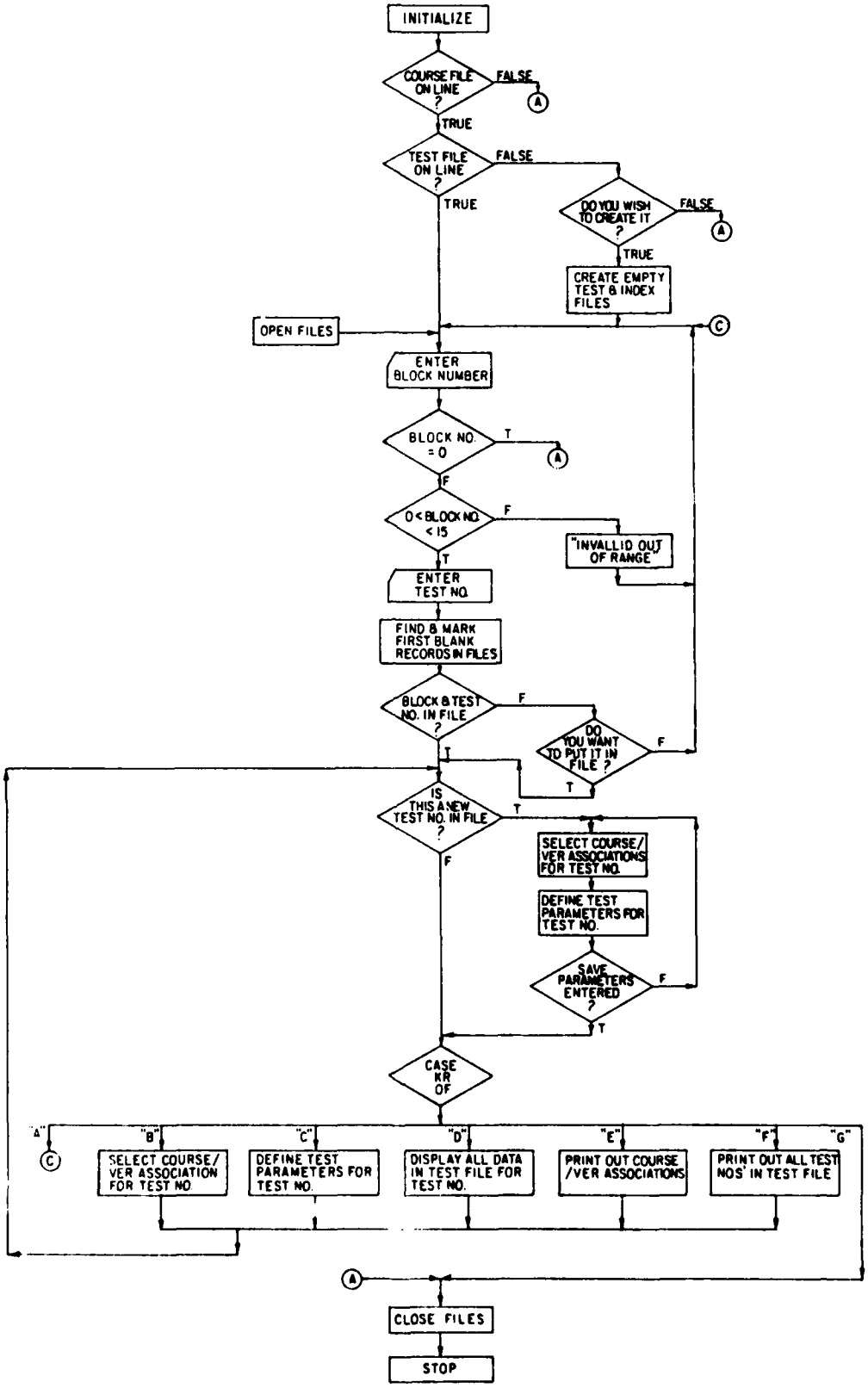


FIGURE C-7. TESTEDT

1. the Student Index File and the Student Data File;

2. the Test Index File.

The initial use for this program is to assign examinations to students. It can be executed only after the Test Editor program has been run to create the examination. The limit on the number of digits/characters allowed for operator data input is:

Student Identification Number: 9 digits/characters.

Note: Only one test can be assigned at a time. After a test is assigned to a student, the assignment can be changed, but it cannot be deleted so as to give no assignment to the student.

The following information can be obtained by calling up this program:

1. the Student Identification Number;

2. the Student Name;

3. the Course/Version and Block in which the student is enrolled;

4. tests assigned, tests not assigned, and tests graded.

A flow diagram for the Assign Test program is shown in Figure C-8.

MICROTERMINAL SOFTWARE

(EROM "FIRMWARE")

The microterminal software used by the students is designed to support student testing procedures in conjunction with the Apple microcomputer software. For operational functions, consult the Student Microterminal Instructions in Appendix A and the written description of the program flow, which accompanies the system flow diagram, Figure C-9 and Figure C-10.

The limits on the number of digits and characters allowed for operator (student) data input are:

1. the Student Identification Number: 9 digits;

2. the Booklet Number: 11 digits;

3. Question Answers: single alpha character.

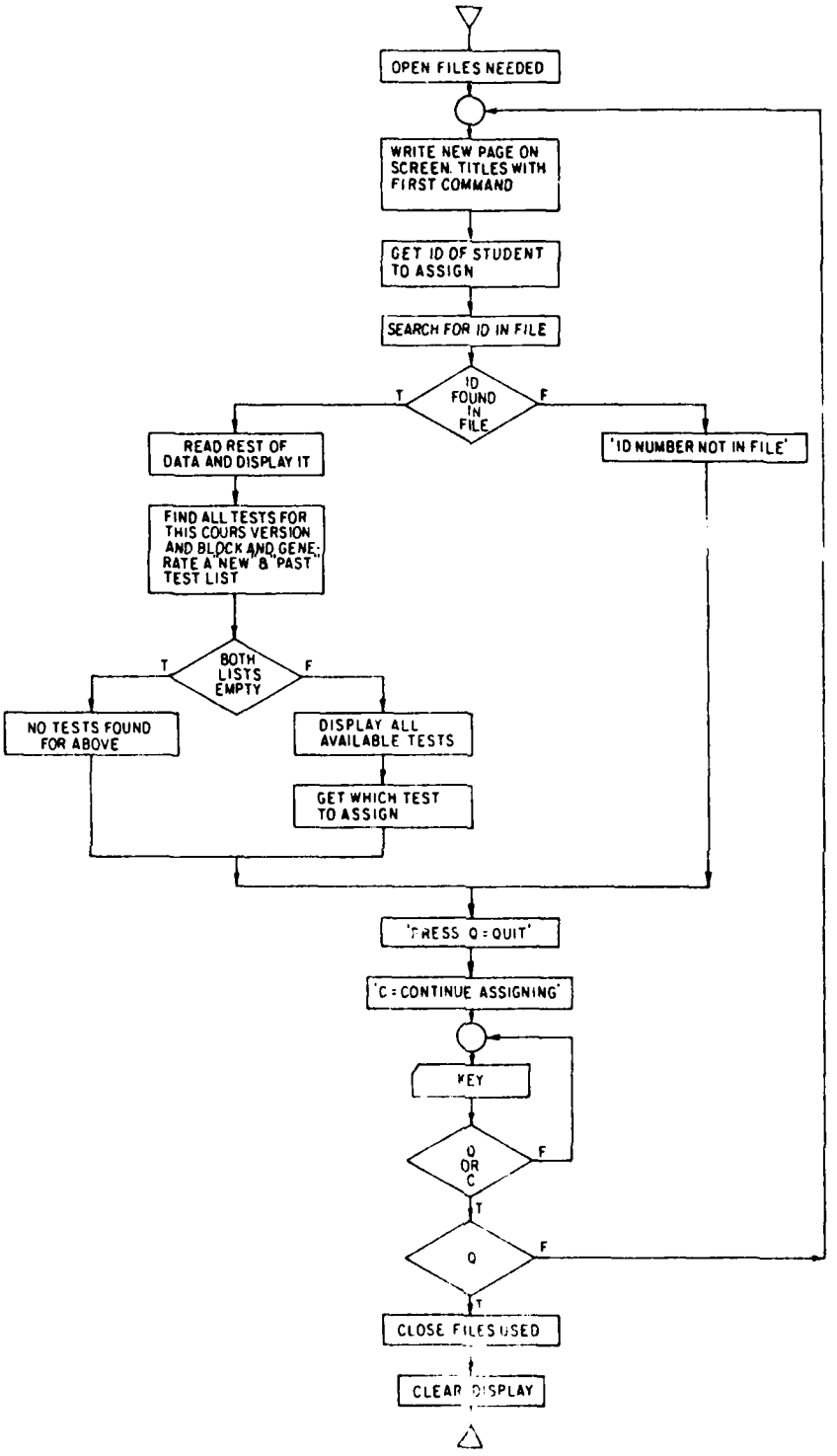


FIGURE C[illegible] ASSIGNTEST

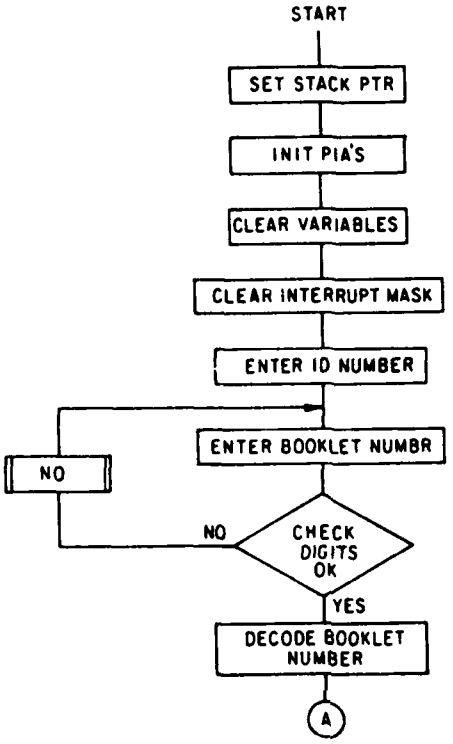


FIGURE C-9. MT30 + SCR (MICROTERMINAL SOFTWARE)

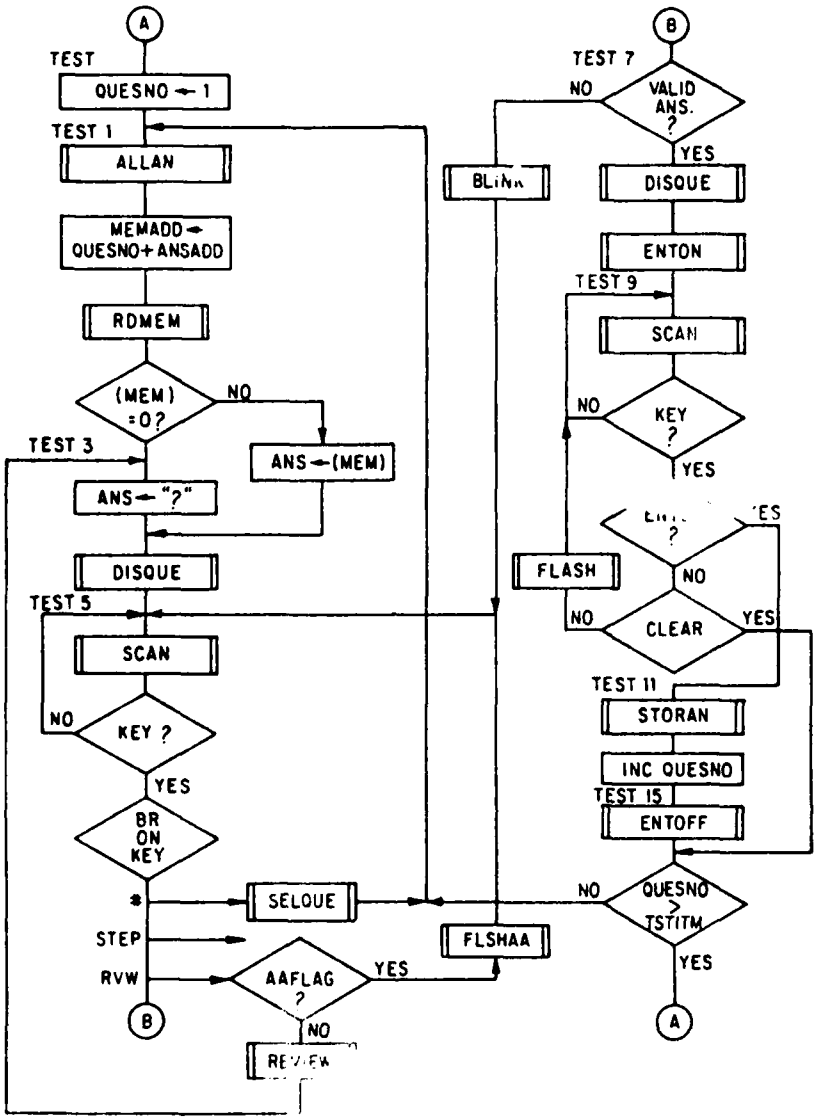


FIGURE C-10. MT30* SCR (MICROTERMINAL SOFTWARE)

The following information is provided to the student through the use of this software:

1. a display of the Question Numbers and "?"'s for unanswered questions (graphic);

2. a display of the Question Number and answer character for answered questions (graphic);

3. "ENTER SSN" message (LED light);

4. "PRESS ENTER" message (LED light);

5. "ENTER BOOKLET NUMBER" message (LED light);

6. "NO" message (graphic);

7. "ALL QUESTIONS ANSWERED" message (LED light).

The following information describes program flow and is written as an accompanying narrative to Figure C-9 and Figure C-10:

The program is stored in two 2716 erasable read only memories (EROM's) in the microterminal. It starts running when the power is turned on by vectoring to the address stored at FFFEH and FFFFH.

It initializes only the peripheral interfaces adapters, sets the stack pointer, and clears all variables.

The "ENTER SSN" message is turned on, and cursors are displayed in all display positions. The cursors remain on until an active memory module is inserted into the microterminal. (An active memory module will have a "1" or a "2" stored at Address "0".)

Upon insertion of the memory module, one of two actions will occur.

1. If the flag at Address "0" is a "1", it will be changed to a "2", and the rest of the memory will be cleared to "0". This will be the case whenever a student is assigned a booklet number and is issued a memory module for testing purposes.

   The student can now enter the SSN. If a digit is entered incorrectly, it can be erased by pressing "CLEAR" and entering the correct digit. After nine digits have been entered, the "PRESS ENTER" LED light will come on to instruct the student to actually enter the number into the memory module.

   The "ENTER BOOKLET NUMBER" LED light will come on and the student will enter the assigned Booklet Number. Two check digits are included in the 11-digit Booklet Number to insure

correct entry. If an error has been made, the "NO" message will be displayed.

Pressing any key will restart the "ENTER BOOKLET NUMBER" routine. After the Booklet Number has been correctly entered, it is decoded to determine the number of questions in the test and the number of choices for the answers. The elapsed t[illegible] counter starts, and the testing begins with Question #1.

2. If the flag at Address "0" of the memory module was a "2" at the time of insertion of the module, the cursors are turned off, and the testing proceeds at the point where the memory module was previously removed.

The test procedure allows for skipping questions by pressing "NEXT", finding the next unanswered question by pressing "RVW", or selecting the next question be answered b[illegible] the ">" key or the "<" key and entering the desired question numbe[illegible].

The display will indicate the question number, followed by an "=" and the stored answer. If the stored answer is "0" (the question has not been answered), a "?" is displayed in the space for the answer.

Answers are entered by pressing the desired key, followed by pressing the "ENTER" key in response to the "PRESS ENTER" message (LED light).

When all questions have been answered, the "ALL QUESTIONS ANSWERED" message is displayed via the LED light, but review and changing of answers is still permitted.

A memory module may be removed and reinserted during testing. The elapsed time clock stops when the module is removed, and restarts when the module is reinserted. The accumulated time is the total of the time actual testing is in progress.

TEST GRADING PROGRAM (GRADETEST)

The Test Grading program allows the operator to grade a student's completed examination and to store the results in the Test Index File, the Test Data File, and the Student Data File.

After the program has been initialized by the operator, the program delays further execution until an "active" memory module is inserted into the transfer microterminal (a microterminal which is connected to the microcomputer). At this point, the operator can choose to grade the test or, if desired, zero out the contents of the module so it can be used again, without grading. If the module is to be graded, the program loads the contents of the memory module into the computer's buffer, and

the buffer downloads the data back to the module, with the first byte set so the module is in an "initialized" condition.

The program then checks for the correct Booklet Number. If the Booklet Number is incorrect, an error message will be displayed, and the program will reset to the "delay" condition for the insertion of an "active" memory module. If the Booklet Number data are correct, the program grades the test, and updates the Test File, the Course File, and the Student Records.

As a provision for future program enhancement, there are "hooks" left so the program, via the "status" condition, can flag a branch and add test results to the Test History File.

The files which are used and/or accessed by the Test Grading program are:

1. the Student Index File and the Student Data File;

2. the Test Index file and the Test Data File;

3. the Course File.

The initial use of this program must follow the execution of the Test Assignment program, where tests are assigned to the students. Because the operator can enter only single-key stroke, menu-oriented responses, there is no concern over digit and character data-entry limits.

The following information can be obtained by calling up this program:

1. Student Identification Numbers;

2. Booklet Numbers;

3. Student Scores, consisting of:

    a. the correct answers/the number of questions on the test;

    b. the percentage of correct responses;

    c. the ratio of the student's scores to the average for the test (compute thus: the sum of the student's current block scores/the sum of the average block scores for the course);

    d. a list of incorrect answers.

A flow diagram for the Test Grading program is shown in Figure C-11 and Figure C-12.

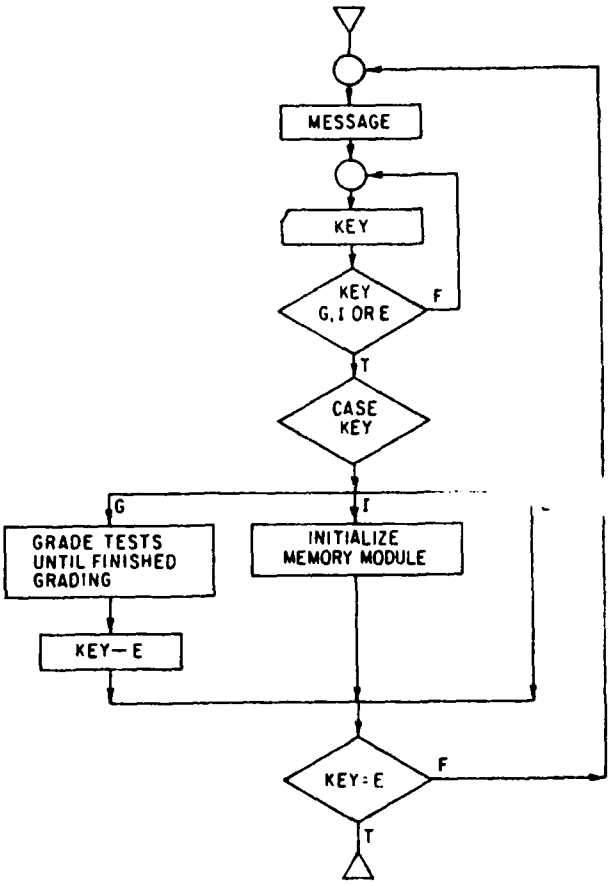


FIGURE C-11. GRADETEST

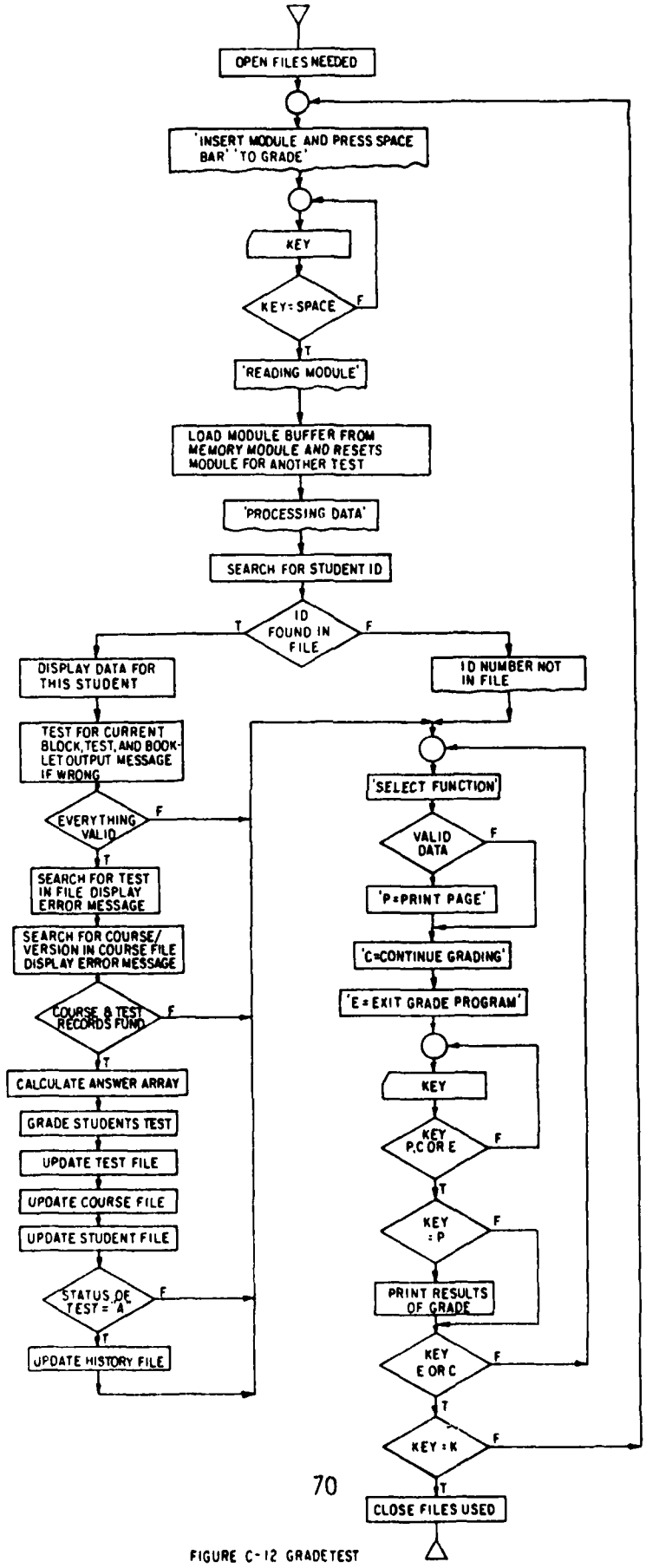


FIGURE C-12 GRADETEST

MICROFICHE-BASED TESTING

The following material is of interest only if the MT/MF system is to be interfaced with the AIS for purposes of microfiche-based testing.

EDITFICHE is implemented via the CAMIL (Computer Assisted, Managed Instructional Language) programming language to complement the various components of the AIS. EDITFICHE must be used in tandem with an AIS software program called EDIT, which can create a text file and fill it with text. This text can then be formatted and positioned within a frame through the use of EDITFICHE. Then, text paper copy can be generated for one or more frames, and a tape can be created to generate Computer Output Microfiche (COM). For purposes of conceptualization, a ready copy of these testing materials is called a fiche, which is intended to be the end product at a future date. The program reads source text from a test file created via the AIS Source Editor (EDIT), formats it, and allows the user to specify where a block of formatted text is to appear within a frame, i.e., allows the user to "move" blocks of text around within a frame.

Once a fiche has been formatted via EDITFICHE and has been actually produced as a microfiche, another software/firmware package developed during Phase II of the effort is utilized. It is at this stage that the microfiche can become the test with which the student will interact. The programs reside in microterminal firmware and on software developed for the Apple microcomputer. This system is referred to as the microterminal/microfiche/Apple system. The instructor interacts with the Apple, using the software to assign tests to student and to record student data. The student uses the microterminal which is yoked to a microfiche reader in order to take tests.

The grading microterminal in this application is controlled by the Apple through the communications interface and is used only to read from and write to a memory module. There is no input on the grading microterminal keyboard required and no messages are displayed. The program for the microterminal is named APPLE * SRC Vol. 1 and is programmed into a single 2716 EROM in the microterminal. The student microterminal is controlled by a firmware program called FICHDSRC which is discussed later in this section of the appendix.

Three programs comprise the software for the Apple microcomputer. All three are written in PASCAL and are stored in diskette files. The first of these programs is named INITANS. It is used by the instructor to initialize the answer arrays in a diskette file named ANSWER FILE. Provisions are made for three test versions of 30 questions each. The second program named INITFRAME is also used by the instructor to initialize another diskette file named FRAMEFILE. This file depicts the physical location of the questions on a microfiche film and is used by the main program during the initialization of a memory module. The third program is titled COMBINED and is the main program used by the

instructors to initialize memory modules and to read and file student test results.

Program COMBINED will automatically be executed by inserting the diskette into Drive 1 and turning on the Apple power switch. Ten functions are displayed on a menu and numbered 1 through 10. When a function selection > 10 is entered, an "INVALID ENTRY" message appears, followed by the menu page again. If the entry is > 99, the program aborts and the computer reverts to its command mode. The INITANS or INITFRAME programs can then be executed. Rebooting the disk will restart the COMBINED program. Student file data will be intact and the testing session can be resumed.

Execution of this program (COMBINED) results in the menu being displayed on the monitor screen as follows:

SELECT FUNCTION

1 = Upload memory module to buffer
2 = Read buffer data
3 = Change data in buffer
4 = List student File ID numbers
5 = Download buffer to memory module
6 = Remove a student file
7 = Recall a student file
8 = Initialize memory module
9 = Initialize the files
10 = Display frame sequence

The buffer is a 128 byte array in memory that is used for the transfer of data to/from the memory module.

The purpose and operation of each function follows:

1. Upload memory module to buffer

This function will transfer the memory module data to the internal buffer. Several seconds are required for this transfer. The message 'UPLOADING MEMORY MODULE...' will be displayed during the transfer, and upon completion the menu message will reappear. The data from the memory module will also be recorded in the student files.

If a file exists for the current ID number, the new data will replace the old data. If no file exists, one will be created for this ID number. The file is structured to accept up to 30 students at one time. A separate file is maintained that holds all the active ID numbers.

2. Read buffer data

Selection of this function results in the following message being displayed on the monitor:

SELECT DESIRED DATA

1 = Memory flag
2 = ID number
3 = Booklet number
4 = Answers
5 = Score
6 = Time
7 = Return

The instructor can now select which data to review. Function 1 will show the code number at address 0 in the memory module. The codes are:

1 = Memory module initiated for first try
2 = Student has completed first try
3 = Memory module initiated for retest
4 = Student has completed retest
5 = Memory module initiated for critique
6 = Student has completed critique

Functions 2 through 5 are self-explanatory. The series of 1s and 0s on the second line of the score display shows which answers were correct and which were wrong. A 1 denotes a correct answer and the flags are printed in sequence. Function 6 is the elapsed time the student required to complete the test. The readout is in hours: minutes: seconds. Function 7 returns the program to the menu page.

3. *Change data in buffer*

This function allows the instructor to change the memory flag, ID number or booklet number in case of an erroneous entry or other reason. It changes the buffer only. If the changes are to be transferred to the memory module, the "DOWNLOAD BUFFER" function must be called.

4. List student file ID numbers.

This function will print out all of the active ID numbers presently in the student files.

5. Download buffer to memory module

This function is called to load the present buffer contents into a memory module which is plugged into the attached microterminal. This operation requires a few seconds and the message "DOWNLOADING BUFFER..."

will be displayed during this time. Upon completion the menu message will reappear.

6. Remove a student file

This function will remove a single selected student record from the student file and the ID file.

7. Recall a student file

This function will transfer the contents of a selected student's file from the diskette file into the buffer. The diskette file remains unchanged.

8. Initialize memory module.

This function is used by the instructor to initialize a memory module prior to a student taking a test. The appropriate prompting messages are displayed for entering the ID number and the booklet number into the buffer and selecting the testing mode.

The booklet number must be six digits. The first digit is the test type. It must be a five or a six. The differences in test types are discussed later in the MT/MF program instructions. The next two digits are the test version number. This number is used in test type five only. If the test type is six, the two digits can be any value, but they must be present. The fourth and fifth denote the number of questions in the test. This can range from 01 to 30. The last digit is unused in this version but, again, it must be present.

Selection of "FIRST TRY" test mode results in clearing all of the answer array in the buffer module and the generation of the frame sequence that the student will be directed through during the test. Using the ID number as the seed, a random number generator sets the criterion for the selection of questions available on the three tests placed on the microfiche. The address 0 location in the buffer will contain a 1 as previously discussed. When all data have been entered into the buffer, the "CONTINUE" message will be displayed. At this time the sequence can be aborted by pressing any key except "C". If "C" is pressed the buffer is downloaded to the memory module and it is ready for the student to proceed with the test.

Selection of "RETEST" results in the same sequence, except the previous test results are obtained from the disk file for that student and the answer array is cleared and preset accordingly. A correct answer to a particular question on the first try results in a "P" being entered into the answer array for that question. Incorrect first answers result in 0 being entered. The frame sequence that is generated has a fixed offset from the one that was generated the first time. This

insures that the student will be given a different question during the retest. A "3" is placed in the address 0 code position.

The "CRITIQUE" function is the same as "RETEST" except that a "5" is placed in the address 0 code position.

The "frame sequence" that is generated for all three test modes unique for each student. A pseudo random-number generator is called that uses the student's ID number as the seed.

9. Initialize the files

This function is used to clear all disk files and prepare for the beginning of a test session. The question "CLEAR ALL FILES" must be answered before any action is taken. This feedback is required to prevent inadvertent erasure of files.

10. Display frame sequence

This function is a graphic representation of the frames that have been selected for a particular student's test. All of the data should be assembled in the buffer before calling this function. It does not need to have been downloaded to the memory module however. It allows an overall view of what the frame sequence will be during the test.

Student Microterminal Firmware

The MT/MF program is named FICHDSRC. After a student has plugged in a pre-loaded memory module, the pressing of any key will start the test procedure. One of two types of testing is provided depending upon the test booklet number entered into the memory module by the instructor. These tests are called Type 5 and Type 6. See Figure C-13 for a flow chart of FICHDSRC.

Test type 5 presents the exam in a format similar to that of a paper-and-pencil test. The test number embedded within the booklet number determines which one of three different exams on the microfiche will be presented to the student. Test numbers 4, 5, and 6 are used in this version. There are four horizontal LEDs on the microterminal, three of which are used to aid the student in positioning the platen. Test number 4 lights the left LED, and the student knows to view the left one-third of the fiche, test number 5 the middle LED for the center one-third of the fiche, and test number 6 the right LED for the right one-third of the fiche. Thus the student knows which third of the fiche to use. Frame numbers are not used for this test type because the entire one-third of the fiche is available for viewing as if it were a paper test.

FICHDSRC

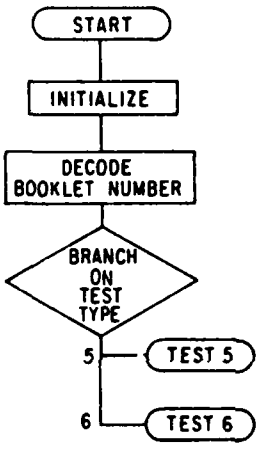


FIGURE C-13. FICH DSRC flow chart

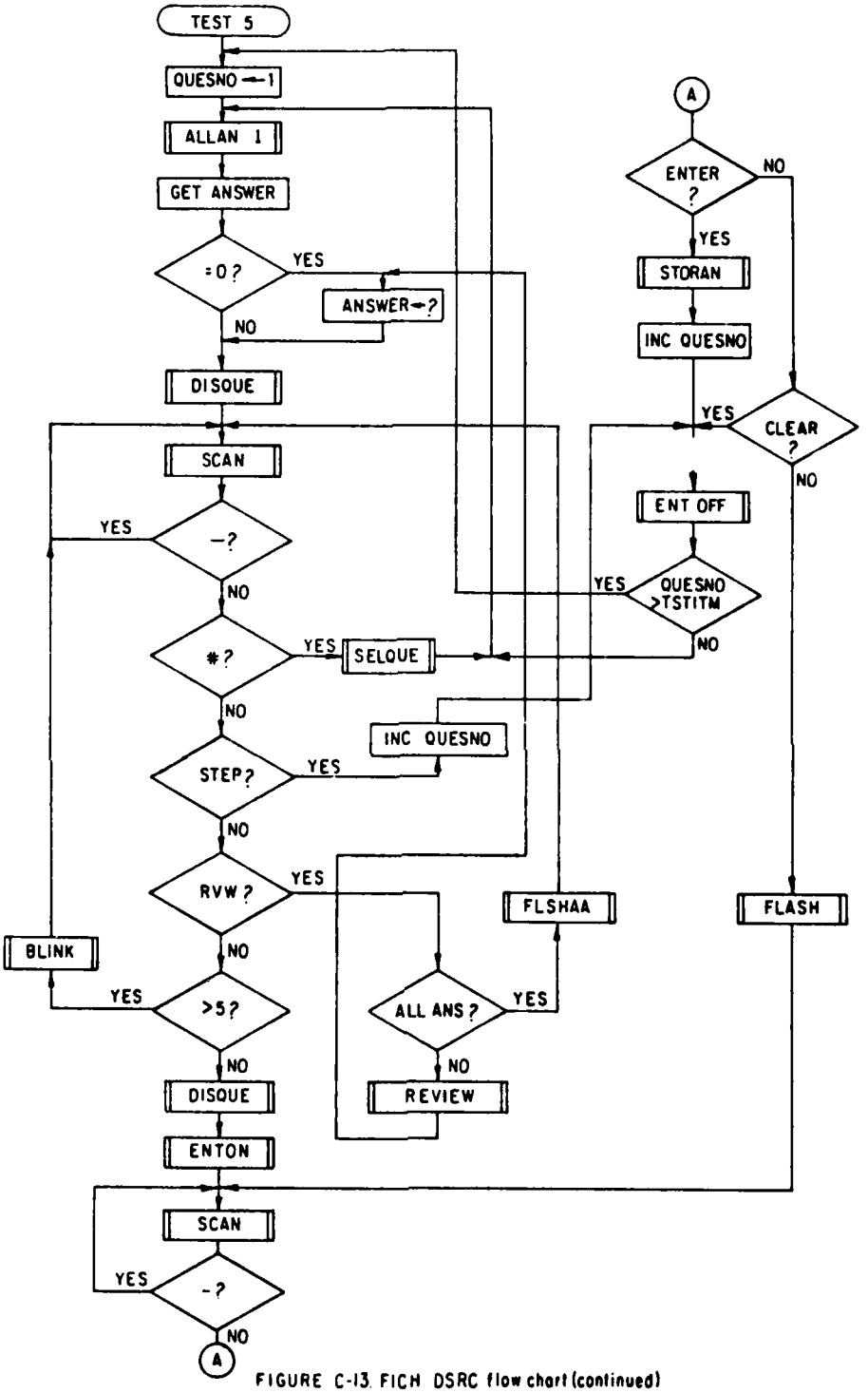


FIGURE C-13. FICH DSRC flow chart (continued)

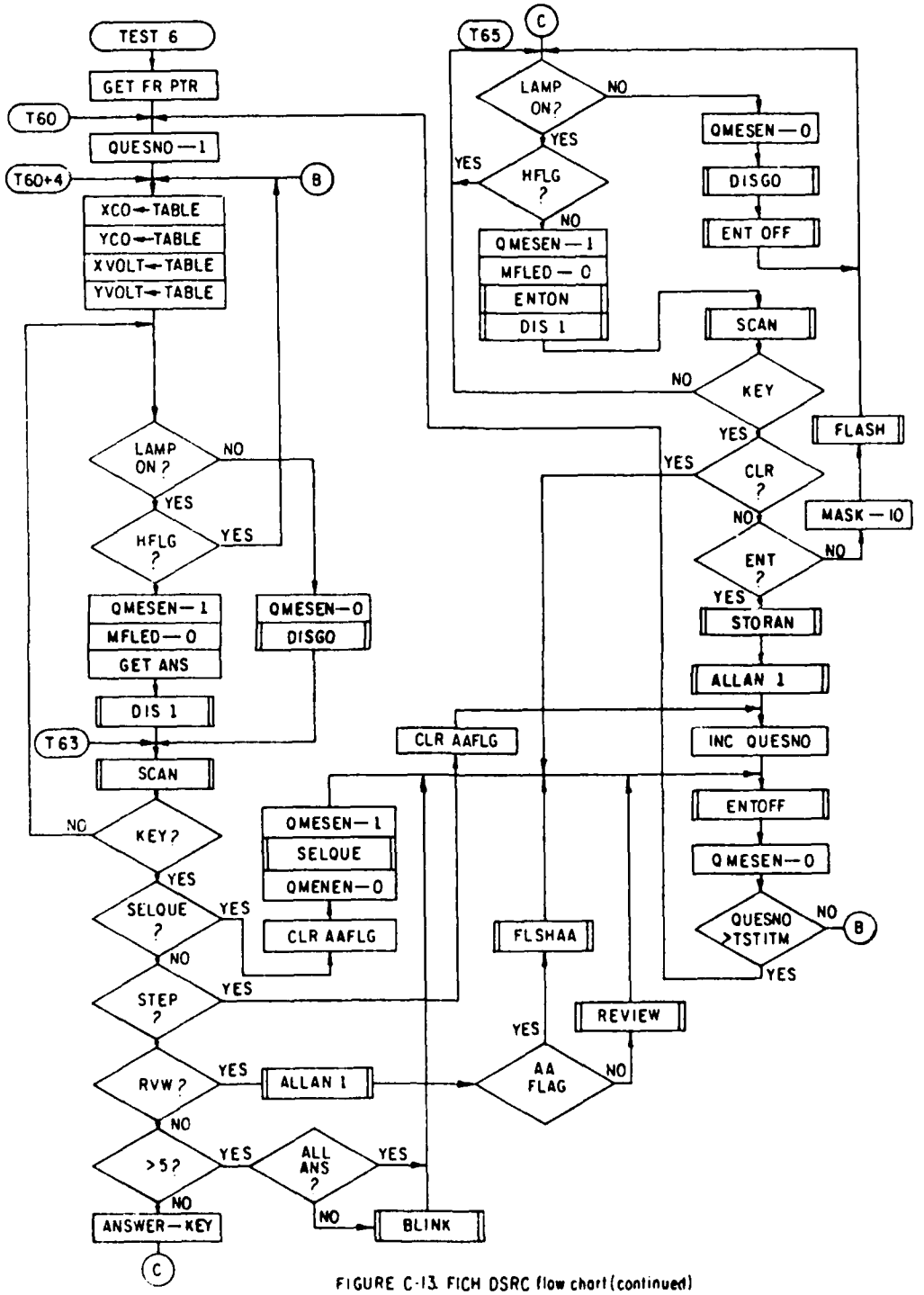


FIGURE C-13 FICH DSRC flow chart (continued)

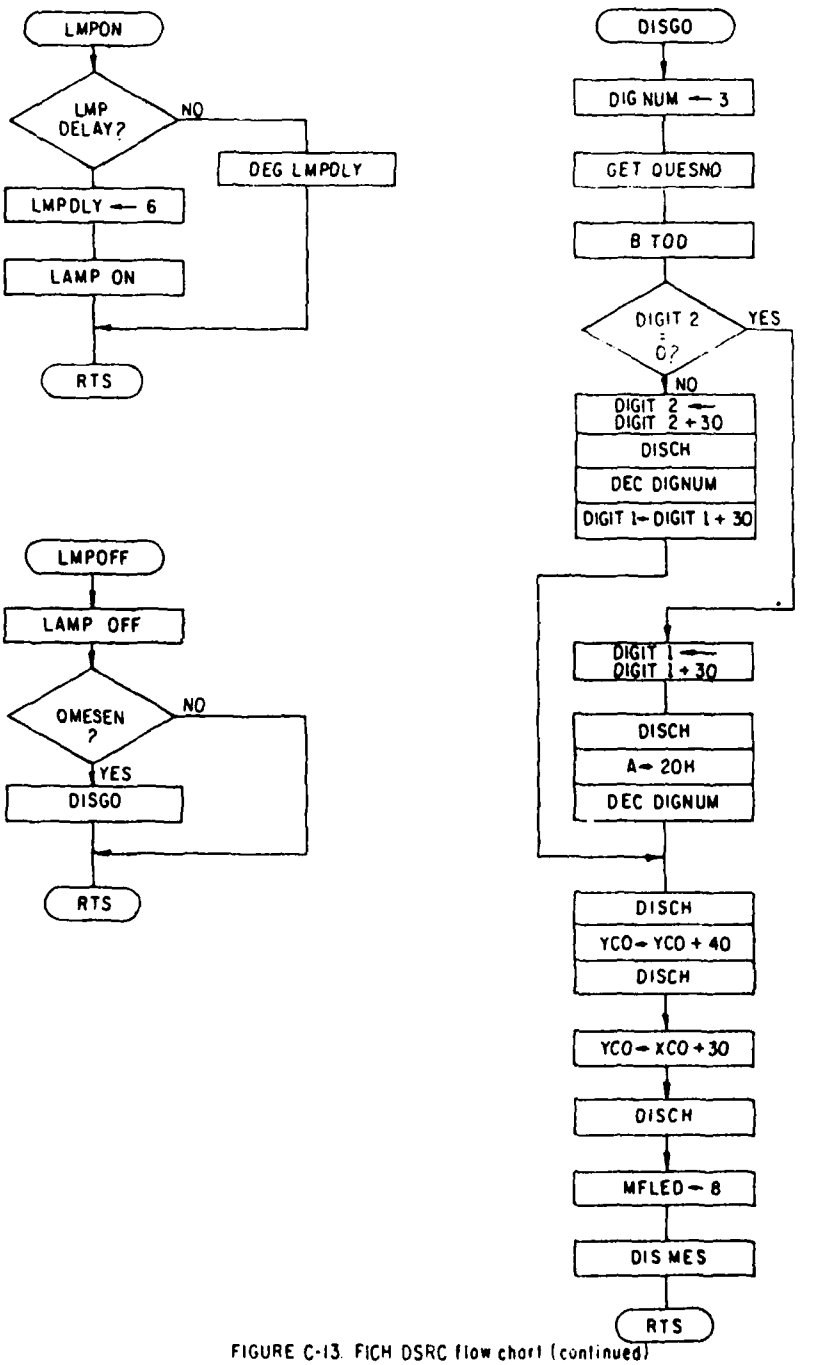


FIGURE C-13. FICH DSRC flow chart (continued)

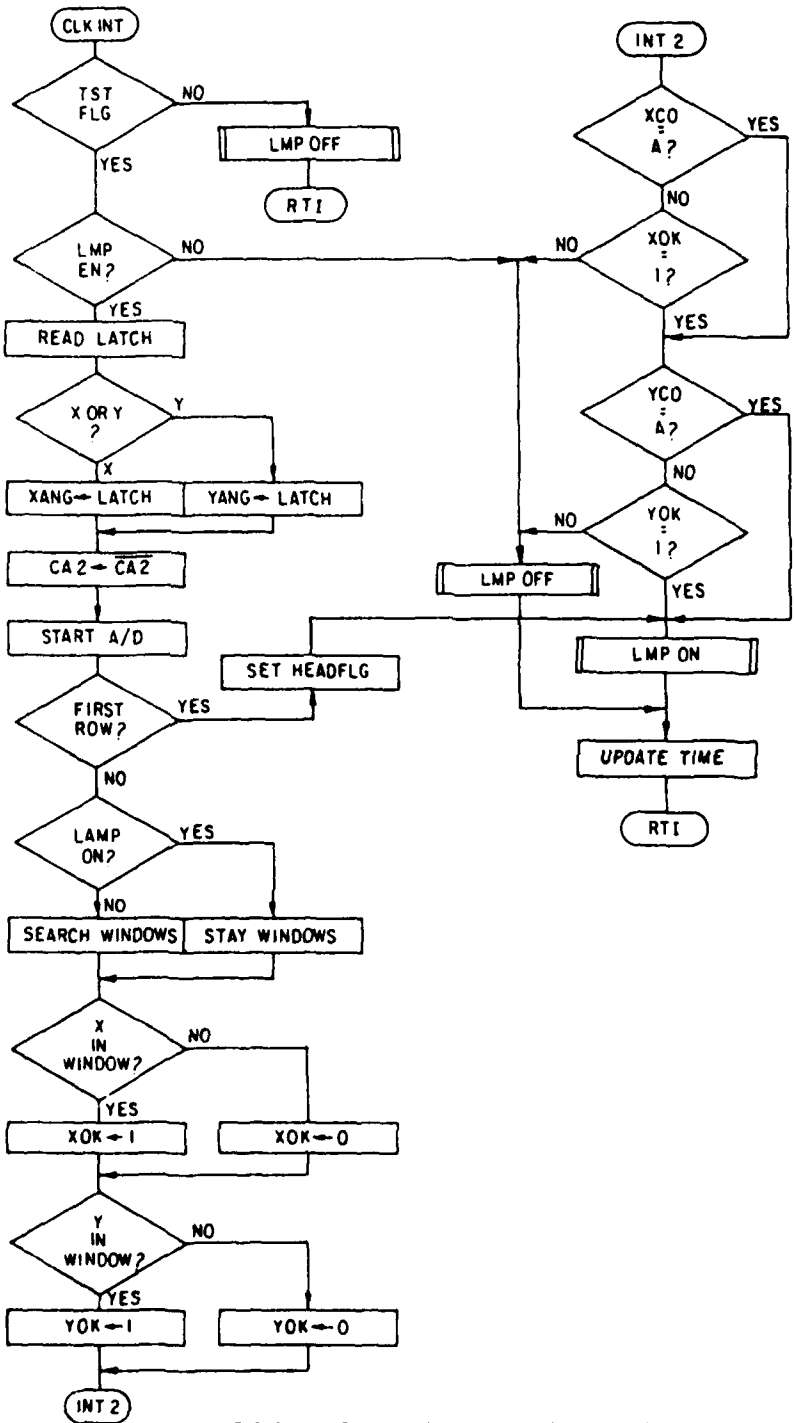


FIGURE C-13 FICH DSRC flow chart (concluded)

Test type 6 is presented to the student on a frame-by-frame basis. The student ID number that was entered earlier into the memory module by the instructor generates a particular "frame sequence" for the student's test. This frame-sequence list is stored in the memory module with the ID number, booklet number, etc. After the student has pressed a key to start this test type, the first display is the frame address that will be used for the first two questions. The fourth horizontal LED, whic shows the status of the carrier, is on when the frame message is displayed in the micro-terminal display and is off when the carrier is properly located by the student.

The program begins running at power on. After initializing the peripheral interface adapters, the program tests the memory module flag at address 0 repeatedly to determine if a module has been inserted. Upon insertion, two courses of action can occur. If the data at address 0 are equal to 1, the data are changed to a 2, and the rest of the module is cleared. This occurs whenever a student is issued a memory module and begins a test. The program goes to START again and the booklet number verification stage is initiated.

A test is made of the booklet number (entered into the module earlier by the instructor) to help insure a correct entry by the use of a check digit. Each digit of the booklet number is multiplied by its position in the whole number and summed to determine the check digit value. Then the number is decoded to determine the test type number, the version number, and the number of questions in the test. The time counter is enabled, and the actual testing begins.

If the flag at address 0 of the memory module is a 2 at the time of insertion, the program proceeds from the point at which the module had been previously removed.

Both test types 5 and 6 provide the ability to skip questions, to sequence through the answer array to locate the next unanswered question, and to allow the selection of the next question to be answered. A review of all test answers is also permitted, and the student can change these answers at will.

A subroutine is included to facilitate the alignment of the carrier. It is accessed by inserting a memory module that has previously had the flag at address 0 set to hexadecimal 12. The lamp then is illuminated for all positions of the carrier, and the display continuously shows the X and Y voltages being returned from the analog to digital converter.

APPENDIX D: RANDOM NUMBER GENERATOR

The test key editor generates randomized tests using a random number generator which has a seed implicit in the "test booklet number." This is so the microterminal tests can be graded off-line using the instructor terminal, or, in a smaller configuration, using a microcomputer system. The primary intention is to use the instructor terminal to provide student pass/fail information and critiques when the central system is not available.

The proposed pseudo-random number is a function x(n,a,b,c,k) where:

n is the test question number,
a is a 16-bit number (fixed as 4360. = 2**12 + 2**8 + 2**3),
b is an 8-bit number,
c is an 8-bit number,
and k is an 8-bit number.

The numbers "a", "b", "c", and "k" form the "seed" for the random number generator. The degree of randomness appears to be sharply dependent on the value of "a", so it has been permanently fixed; "b", "c", and "k" are to be derived from the "test booklet number."

The test booklet number is required to contain all of the information required in the test number on the paper forms (i.e., the block number and the test number) for use by the grading terminal; it is also required to contain the number of questions on the test and the number of items in each question (which is the same for all questions on a given test) for use in the microterminal. In addition, since the microterminal uses the test booklet number to determine the way in which the test is to be administered, check digits are required to be appended to the test booklet number to help prevent errors in keying the test booklet number into the microterminal. Finally, the test booklet number is used to form the seed for the random number generator, so it must be unique for each test version, with some provision for allowing the instructor to vary the key if one is not satisfactory.

The test booklet number consists of 11 digits in two groups, as follows:

b1 - the most significant digit of the block number,
b2 - the least significant digit of the block number,
t1 - the most significant digit of the test number,
t2 - the least significant digit of the test number,
ck1 - check digit for the first data group
n1 - the next-to-least significant digit of the number of questions on the test,
n2 - the least significant digit of the number of questions on the test,

u - the number of items on each test question,
v1 - the most significant digit of the test randomizer,
v2 - the least significant digit of the test randomizer,
ck2 - check digit for second data group.

The check digits are formed as follows:

```
ck1 = MOD(         2*b1 + 7*b2 + 3*t1 + t2; 10 )
ck2 = MOD( 9*n1 + 2*n2 + 7*q  + 3*v1 + v2; 10 ).
```

The test booklet number is the sequence of digits given above, in the order shown, with a space separating the two groups.

The random number seed is derived by equating the number formed by concatenating the 8-bit numbers "b", "k", and "c" (with "b" most significant and "c" least significant) with the 24-bit number which is the least significant part of the decimal number which may be written (from most- to least-significant digits) as:

```
v1.v2.b2.a.n2.n1.t2.b1 + v1.v2
```

where the number "v1.v2" is added into the result using ordinary arithmetic, in order to allow a degree of control over the low order part of the result, as well as the high order part. Additionally, "b" and "k" are incremented, if necessary, to make them odd numbers. This is done to avoid having a random sequence with a period of less than 256.

E.g., using an intermediate variable, z:

```
z = 10*(10*(10*(10*v1+v2)+b2)+a)+n2
z = 10*(10*(10*z+n1)+t2)+b1 + 10*v1+v2
z = MOD( z ; 2**24)
b = MOD( z DIV 2**16 ; 2**8 )
        IF (MOD(b,2) .EQ. 0) b = b + 1
k = MOD( z DIV 2**8 ; 2**8 )
        IF (MOD(k,2) .EQ. 0) k = k + 1
c = MOD(z ; 2**8 )
```

The random number generator is then

```
x(n ; a,b,c,k) = MOD( # a*m # & count(MOD(b*m,256)) ; Q)
```

where

```
m = babble( MOD( k*n+c ; 2**8 ) )
```

babble() is a "bit-babbling" function (scatter the bits, then scoop them up in a different order), as follows:

If the bits of an 8-bit integer are numbered from 1 (least significant) to 8 (most significant), then:

bit 1 of the output is bit 2 of the input,
bit 2 of the output is bit 4 of the input,
bit 3 of the output is bit 1 of the input,
bit 4 of the output is bit 6 of the input,
bit 5 of the output is bit 3 of the input,
bit 6 of the output is bit 8 of the input,
bit 7 of the output is bit 5 of the input,
bit 8 of the output is bit 7 of the input.

#...# represents the operation of taking the center 8 bits of a 24-bit number (i.e., MOD(... DIV 256 ; 256);

& represents the exclusive OR operation (bit-by-bit on fixed format integers).

Count() represents the bit-counting function. It has the value (0..8) of the number of bits in the 8-bit number which are a "1".

The ranges of the various data items are indicated below:

	DATA item	AIS range	microterminal range.
	course number	(1..999)	not used
	course version	(1..31)	not used
b1.b2	block number	(1..15)	(1..15)
t1.t2	test number	(1..20)	(1..99)
	test version	(0..127)	not used
n1.n2	number of questions	(1..144)	(1..50)
Q	number of items	(2..5)	(2..5)
v1.v2	test randomizer	not used	(00..99)

In cases where the microterminal and AIS ranges differ, the smaller of the two should be used, except that the number of questions for the microterminal is an operational restriction: the appropriate range is (1 through 99), except that values over 50 may not be used with the microterminal until the capacity of the microterminal memory module is expanded.

The number of questions (n1.n2) are treated as the least significant two digits of the number of questions in the AIS system. For tests with more than the number of questions allowed by the microterminal, the test must be administered via other means (e.g., on-line terminal or paper form), and the presence of the number of questions in the booklet number is not significant. The random number generator x(n; a,b,c,k) does not use the maximum number of questions (except implicity in the seed), and is valid for n in the range (1..256). Hence, the AIS system

can continue to use the test booklet number for tests within its normal range, if desired.

Note that the course number and version are not included in the test booklet number since they are not required by either the grading-terminal program or the microterminal program.

In formulating the seed, an attempt has been made to ignore the high-order digit of items with a restricted range, since that digit will not change very much, and the intent is to generate a seed with as much variability as possible.

In addition to the random number generator, it is recommended (although not required) that statistical tests of randomness be applied to the test keys generated using the proposed generator. A display of the test key could be substituted for the mathematical tests, with the instructor making the choice.

The recommended tests are:

1. Frequency of occurrence testing: the number of A's, B's, C's, etc., should be approximately equal.

2. Successors testing: The probability of a particular answer should not be dependent on the answer to the previous question.

3. Length of runs: A "run" is a monotonic sequence of numbers (here, the numbers are equated with A=0, B=1, C=2, D=3, E=4, etc.) The length-of-runs test is predicated on a tally (for each point) of the length of the run which includes that point. (Note that every point except the first is a member of either a run-up or a run-down of length at least one.)

This pseudo-random number generator generally passes the frequency-of-occurrence test and the successors test, but performs poorly with the length-of-runs test. This version of the random number generator has been tested on a limited sample of test keys. The sample consisted of all possible randomizer values for five combinations of block numbers, test numbers, number of questions, and number of items (a total of 500 keys). Over half of this sample passed all of the statistical tests when the computed statistic was required to be in the 1% to 99% range of its expected range of values. Some problems were observed with the items per question equal to 2, so a comparison was run (on 200 keys) using the DEC FORTRAN random number generator. Only one key was satisfactory within the same limits. When the length-of-runs testing was ignored, this rose to about 90% satisfactory for the DEC generator, and about 95% satisfactory for the DRI generator. It should be noted that passing or failing these tests is not a sure-fire technique for selecting suitable test keys: the operator (instructor) should be required to examine any key and to determine if it is suitable.

The statistics were derived from the following sources:

Abramowitz, M., and Stegun, I.A. (Ed.), Handbook of mathematical functions, National Bureau of Standards Applied Mathematics Series - 55 (Issued June, 1964), pp. 949-950.

Hull, T.E., and Dobell, A.R., Random number generators, SIAM Rev. 4 (1962), pp. 230-254.